VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Platforma Microsoft Surface
Microsoft Surface Platform

2010 Marek Kocián

Zadání

Cílem této práce je seznámení se s MS Surface, jeho možnostmi a především s vývojovým prostředím a jeho specifiky.

1. Seznamte se s architekturou, filozofií a celkovým konceptem zařízení MS Surface, a to především z pohledu vývoje aplikací pro tuto platformu. Tyto poznatky popište.

2. Analyzujte možnosti a požadavky kladené na aplikace určené pro toto zařízení. Popište možnosti a specifika vývoje aplikací, a to především s důrazem na GUI a jeho vazbu na komunikační rozhraní a funkce poskytující toto zařízení.

3. Vytvořte kolekci ukázkových příkladů, kdy bude ilustrováno použití specifických možností MS Surface s důrazem na GUI a použitelnost.

4. Analyzujte, navrhněte a implementujte komplexní aplikaci pro podporu prodeje v kamenném obchodě s napojením na reálná data (např. elektro zboží). Aplikace bude ilustrovat možnosti použití tohoto zařízení v reálném provozu se zaměřením na běžné zákazníky. Jako vstup dat do aplikace bude použit obecný formát pro výměnu informací o zboží.

5. Zhodnoťte možnosti aplikací a jejich reálné nasazení na platformě MS Surface. Pokuste se získat názory na nasazení aplikace pro podporu prodeje od běžných potencionálních uživatelů resp. zákazníků.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. května 2010

.....

Marek Kocián

Poděkování

Rád bych na tomto místě poděkoval všem, díky kterým tato práce vznikla. Jmenovitě pak mému vedoucímu diplomové práce Ing. Michalovi Radeckému za rady a podněty k obsahu textu a vyvíjeným aplikacím. Také bych chtěl poděkovat svým nejbližším a přátelům za trpělivost a velkou podporu při studiu.

Děkuji.

Abstrakt

Tato práce pojednává o nové technologii Microsoft Surface. Cílem je popsat její vlastnosti a funkce s ohledem na uživatelské rozhraní. Součástí práce je také vytvoření série ukázkových aplikací, které by měly demonstrovat základní vlastnosti této platformy. Obsahem jedné z nich má být komplexní aplikace pro podporu prodeje v reálném obchodu. Součástí je i zhodnocení možností aplikací a jejich reálného nasazení na platformě Microsoft Surface.

Klíčová slova

Microsoft Surface, SurfaceShop, tagy, uživatelské rozhraní, kontakty

Abstrakt

This work is about new Microsoft Surface technology. The aim is to describe its features and functions with regard to user interface. The work also includes a serie of sample applications, which should demonstrate the basic features of the platform. The content of one of them has to be complex application for sales support in real shop. There is also an assessment of the possibilities applications and their real setting to the Microsoft Surface.

Keywords

Microsoft Surface, SurfaceShop, tags, user interface, contacts

Seznam použitých zkratek

API – Application Programming Interface

HDTV – High-Definition Television

NFC - Near Field Communications

RFID - Radio Frequency Identification

SDK – Software Development Kit

SP – Service Pack

SQL - Structured Query Language

SŘBD – Systém Řízení Báze Dat

URI – Uniform Resource Identifier

URL - Uniform Resource Locator

WIFI – Wireless LANN

WPF – Windows Presentation Foundation

XML – eXtensible Markup Language

Obsah

Seznam obrázků

1. Úvod

S dotykovými displeji se již jistě setkal každý. Mnoho z nich je využíváno pro účely mobilních zařízení, ale našly si uplatnění i v jiných oborech. Různé druhy displejů se mohou v detailech lišit, ale jedna věc je jim všem společná: okamžitá reakce zobrazovaného obsahu na fyzický kontakt. Uživateli je tak umožněno manipulovat s obsahem mnohem intuitivnějším a zábavnějším způsobem, než jak je tomu u běžných počítačů.

Všechna zařízení tohoto typu mají jednu zásadní nevýhodu. Konstrukce jim nedovoluje rozeznávat více než dva kontakty současně. Konferenční stolek Microsoft Surface však do této kategorie nepatří. Ačkoliv se při zběžném pohledu může zdát, že se jedná jen o další z obyčejných dotykových displejů pojatých netypickým způsobem, opak je pravdou. Ve skutečnosti toto zařízení nabízí mnohem víc, mimo jiné i věci, které až do teď nebyly možné.

Obrázek 1: Stoleček Microsoft Surface

Cílem této diplomové práce je seznámit širokou veřejnost s možnostmi Microsoft Surface. Tato problematika ještě není v obecném povědomí, proto se následující kapitoly budou snažit seznámit čtenáře s hardwarovým a softwarovým zařízením, které se pod názvem Microsoft Surface skrývá, a v rámci možností a zkušeností popsat způsob, jakým se aplikace pro tuto platformu vyvíjejí.

2. Microsoft Surface

Na první pohled vypadá Microsoft Surface jako konferenční stolek, jenž má místo desky dotykový displej. Přistoupivšímu uživateli je tak umožněno manipulovat se zobrazenými daty pomoci doteků prstů, jako by byly data fyzicky přítomny. Jedinečná je tato zkušenost v tom, že na rozdíl od většiny dotykových displejů lze použít i více prstů současně. Díky tomu může uživatel provádět s daty mnohem složitější manipulace, než tomu bylo doposud.

Obrázek 2: Posunutí objektu v prostředí Microsoft Surface

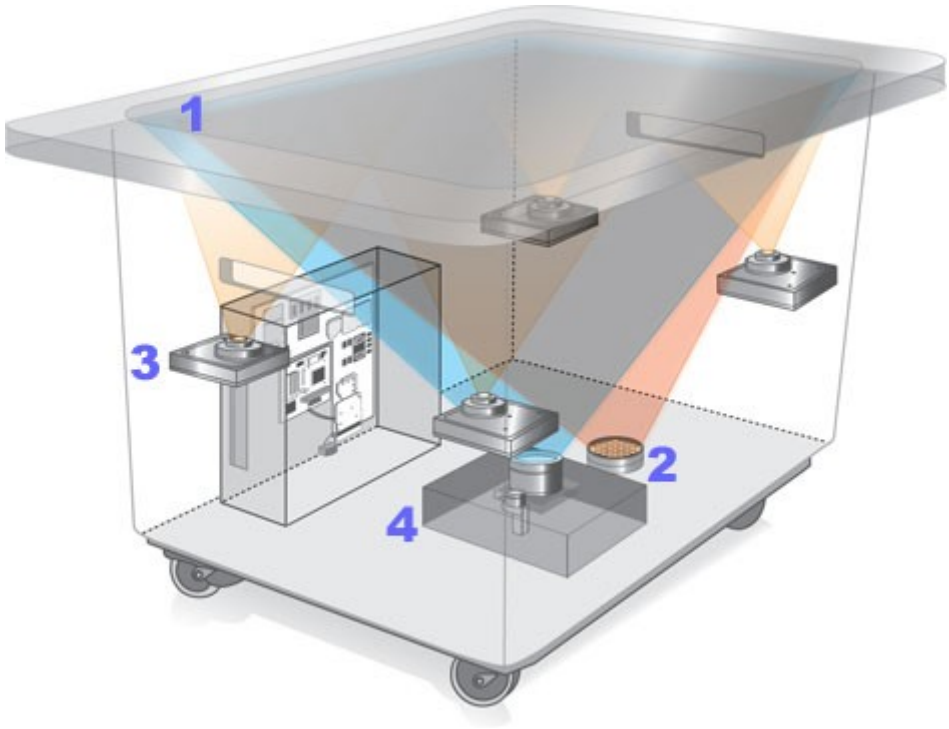
Všechny manipulace uživatele přitom vycházejí z přirozených pohybů, což dělá ovládání mnohem intuitivnějším a snáze zapamatovatelným. Další z charakteristických funkcí, které stojí za to zmínit, je schopnost Microsoft Surface identifikovat předmět položený na desku stolku, například mobilní telefon či kameru. Identifikovaný předmět lze zahrnout do činnosti aplikace.

Obrázek 3: Použití více kontaktů současně pro zvětšení objektu

Díky schopnosti rozpoznávat několik kontaktů současně může nad stolkem pracovat i více uživatelů současně. To otevírá nové možnosti pro tvorbu aplikací a jejich používání. Stolek a jeho schopnosti se dají využít v soukromém i veřejném sektoru. Běžný uživatel tak získá zařízení, které se ovládá zábavným způsobem, a firmy získají neocenitelného pomocníka pro komunikaci se zákazníkem i pro komunikaci mezi sebou.

2.1. Microsoft Surface jako zařízení

Novinka společnosti Microsoft počítač Surface ve tvaru konferenčního stolku postrádá jakékoliv běžné vstupní zařízení. Displej, který plní funkci desky stolku, má úhlopříčku 30 palců (76 cm) a je chráněn plastovým krytem. Základna stolku má 22 palců (56 cm) na výšku, 21 palců (53 cm) na šířku a 42 palců (107 cm) na délku [1]. Celé zařízení váží víc než 100 kg, proto je stolek zespod vybaven kolečky, aby ho bylo možno přesouvat.



Obrázek 4: Schematický nákres stolku Microsoft Surface

Princip celého zařízení je poměrně prostý, jak je vidět na uvedeném obrázku. V základně stolku je ukryt běžný počítač s procesorem Intel Pentium 4 na frekvenci 3,0GHz a 2GB pamětí RAM. Používá 32 bitovou verzi operačního systému Windows Vista. Deska stolku, na obrázku označená číslem 1, je tvořena rozptylovým sklem a svým složením se neliší od běžných dotykových displejů. Velikost desky umožňuje pohodlný přístup i více uživatelům současně. Deska stolu sama o sobě kontakt nesnímá. To provádí pět kamer v základně stolu pracujících na principu odraženého infračerveného světla. Na obrázku jsou označené číslem 3.

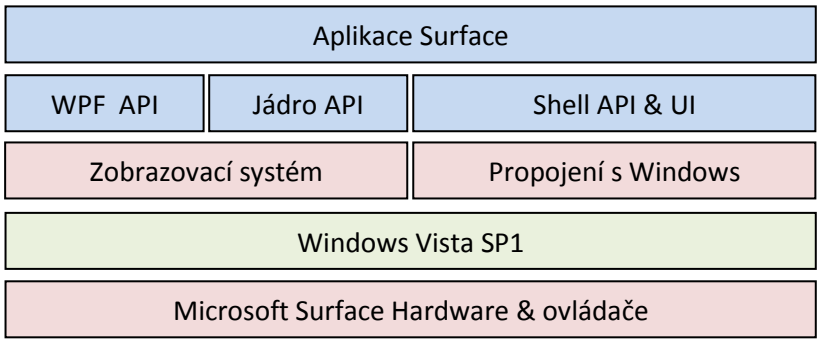
Infračervené světlo o vlnové délce 850 nanometrů je emitováno ze světelného zdroje LED, na obrázku označený číslem 2. Celé snímání probíhá tak, že kontakt s deskou stolu způsobí odraz vysílaného světla do jedné z kamer. Každá kamera přitom snímá jen část povrchu stolu, proto lze rozlišení snímání nastavit na poměrně vysokou hodnotu 1280x960. Díky tomu jsou kamery schopny rozeznat i nepatrný kontakt. Jediným jejich omezením je rychlost zpracování, ale i tak je zařízení schopno rozpoznávat až 52 kontaktů současně [2].

Pro zobrazení informace na displej používá Microsoft Surface stejný způsob projekce jako HDTV projektory. Na obrázku je projektor zobrazen pod číslem 4. Stolek Surface sice pro zobrazení výsledného obrazu používá nízké rozlišení (asi 1024x768), ale díky tomu dokážou kamery lépe rozeznávat hrany obrazovky. Pro spojení s okolím používá počítač Microsoft Surface bezdrátové připojení WIFI a anténu Bluetooth. Do budoucna se počítá ještě s připojením RFID (Radio Frequency Identification) a s NFC (Near Field Communications) [3].

2.2. Software pro Microsoft Surface

Stolek Microsoft Surface nemá po softwarové stránce žádné nároky, protože veškeré potřebné aplikace se nainstalují současně s operačním systémem a uživatel tak dostane stolek plně funkční. Jiná situace ovšem nastává, pokud uživatel hodlá současně vyvíjet vlastní aplikace. Tento požadavek je ze strany společnosti Microsoft svázán s poměrně tvrdými pravidly. Asi největší překážkou vývoji aplikací pro Microsoft Surface je cena zařízení. Podle aktuálního ceníku z prosince 2009 se samotná jednotka pro komerční využití prodává za 12 500 $. Tato cena se však ještě zvýší, pokud uživatel projeví zájem o licenci pro vývoj aplikací. Celé zařízení i s instalačním balíčkem pro Microsoft Surface pak stojí až 15 000 $ [4].

Pro samotný vývoj aplikací není vlastnictví konferenčního stolku Microsoft Surface podmínkou, stačí běžný počítač splňující minimální požadavky. Hardware by měl být minimálně takový, aby na počítači šlo bezchybně provozovat operační systém Windows Vista Enterprise nebo Ultimate opatřený service packem (SP1). Microsoft Surface je optimalizován pro 32 bitovou verzi, proto musí být stejné verze i operační systém počítače. To je poměrně svazující, stejně tak jako fakt, že není podporován nejrozšířenější operační systém této třídy: Windows Vista Home. Před samotnou instalací Microsoft Surface SDK na počítač je třeba ověřit ještě další požadavky na systém, například podpora .NET frameworku nebo možnost vyvíjet WPF aplikace (Windows Presentation Foundation).



Obrázek 5: Diagram obecné implementace Microsoft Surface

Během psaní tohoto textu se na trhu objevil nový operační systém firmy Microsoft: Windows 7. Tento operační systém představuje do budoucna možné odbytiště pro aplikace, které až doposud bylo možno provozovat jen na platformě Microsoft Surface. Windows 7 má totiž také podporu multidotyku (multitouch), což umožňuje používat aplikace určené pro stolek i zde. Podmínkou ale je počítač vybavený monitorem, který multidotyk umožňuje.

Během doby, kdy tento text vznikal, byl vydán i service pack přímo určený platformě Microsoft Surface. Jeho použití je však poněkud problematické. Sice opravuje spoustu chyb a přidává řadu nových komponent, zároveň však upravuje i stávající funkce. Řada aplikací vytvořené v základní verzi Microsoft Surface je tak po instalování service packu nefunkční nebo je jejich funkčnost omezena. Přesto se doporučuje tento service pack instalovat, protože nové funkce mnohem víc práci usnadňují, než komplikují.

2.3. Základní rysy Microsoft Surface

Beze sporu největší výhodou Microsoft Surface je intuitivní ovládání. Konečky prstů se dají informace zvětšovat, zmenšovat, otáčet i posouvat. Takový systém přejde běžnému uživateli do krve mnohem rychleji, protože vychází s přirozených pohybů rukou. Díky své snímací desce stolek nahradí myš, klávesnici, ale i tablet, takže uživatel není ochuzen o jiné druhy interakce. Aplikace pro tuto platformu se obecně lehce ovládají a je jedno, jestli uživatel kreslí, píše či jenom dotekem ovládá komponenty. Vše působí zcela přirozeně.

Obrázek 6: Multidotyk v praxi

Další významnou vlastností, kterou se Microsoft Surface liší od běžných dotykových displejů a které si jistě všimne i běžný uživatel, je způsob, jakým se manipuluje s některými základními prvky. Lze například chytit obraz za protilehlé rohy a tím ho zvětšit či zmenšit. Tyto manipulace mohou být ještě složitější, protože Microsoft Surface dokáže rozpoznávat až 52 současných kontaktů, tedy dost kontaktů na to, aby dva uživatelé mohli použít všechny prsty. Tato vlastnost se také nazývá multitouch (multidotyk). Zařízení navíc dokáže jednotlivé kontakty od sebe odlišit, protože každý kontakt s deskou stolu má přiřazené jedinečné identifikační číslo. Tuto vlastnost má jen málo dotykových displejů a uživatelé ji jistě ocení. Například pokud si uživatel v aplikaci typu Malování vybere barvu z palety, může prstem touto barvou kreslit. Jakmile si někdo další vybere novou barvu a začne jí kreslit, barva prvního uživatele zůstane nezměněna (kontakt má zaznamenané identifikační číslo, na které aplikace navázala původní barvu), ale jen do doby, než uživatel přeruší kontakt s dotykovou plochou. Po té je dotyk prstu vyhodnocen jako nový bod a dostane barvu, která byla v aplikaci navolena jako poslední [5].

Množství kontaktů, které lze současně snímat, poskytuje uživatelům Microsoft Surface nejen možnost používat všechny prsty, ale i možnost pracovat v týmu. Není třeba se nikterak složitě připojovat, nosit si vlastní zařízení, stačí jen přistoupit ke stolku a hned se lze zapojit do dění. Stolek tak lze využít i pro aplikace, kde se očekává přístup více uživatelů současně, například při hře Člověče, nezlob se, a které až dosud bylo možné realizovat pouze pomocí síťového propojení. Právě možnost přístupu více uživatelů dělá práci s Microsoft Surface tak zábavnou.

Mimo běžné kontakty umožňuje Microsoft Surface využívat i objektů umístěných na povrch stolku. Tuto možnost nemá žádný dotykový displej na trhu a dělá tak toto zařízení jedinečným. Tajemstvím této schopnosti tkví v použití tagů. Tag není nic jiného než vytisknutý identifikátor, který se připevní na povrch objektu, jenž má být v aplikaci rozpoznán. Díky němu pak vznikají efektní aplikace, ve kterých například uživatel položí mobilní telefon na desku stolu a počítač v Microsoft mu pak umožní prohlédnout všechny obrázky na něm uložené. Ve skutečnosti však není rozpoznán mobilní telefon, ale tag, který je na jeho povrch připevněn. Pro identifikaci objektů má Microsoft Surface možnost použít hned dvou typů tagů. Prvním typem je Byte tag, jinak také zvaný Domino tag, a druhým je Identity tag. Tyto tagy se liší vzhledem, ale v zásadě je jejich funkce stejná: jednoznačně identifikovat objekt umístěný na desku stolu.

2.4. Rozdíly mezi běžnými aplikacemi a aplikacemi pro Microsoft Surface

Aplikace pro Microsoft Surface se vyznačují několika výraznými rysy, jimiž se odlišují od svých běžných protějšků. Asi nejvýraznější je způsob ovládání, ale ten byl popsán v předcházející kapitole a je patrný na první pohled, proto je zde pouze zmíněn. Ovládání však není jediné, čím se Microsoft Surface odlišuje. Proto autor tohoto textu sestavil stručný seznam dalších rozdílů, kterých si všiml. Seznam se snaží vyznačit hlavně rozdíly mezi aplikacemi pro tuto platformu a běžně používanými aplikacemi.

Většina aplikací pro Microsoft Surface má poměrně jednoduché uživatelské rozhraní. Vždy je základem volná plocha, po niž se pohybují jednotlivé položky a ovládací prvky. Těch není mnoho a většinou jsou již od spuštění aplikace viditelné. Plocha tak nikdy není zcela prázdná, protože se ovládací prvky vyskytují na ní. U běžných aplikací je to přitom obráceně, vývojáři se snaží o to, aby při prvním spuštění byla pracovní plocha volná a ovládací prvky umístěny kolem ní. Druhou používanou možností je, že jsou ovládací prvky skryty a volají se pomocí menu. V aplikacích pro Microsoft Surface se menu pro zobrazení ovládacích prvků

téměř nepoužívá. Když už je potřeba některé ovládací prvky skrývat (protože jsou rozměrné, nepoužívají se tak často), jejich vyvolání je většinou záležitostí tagů a ne menu.

Málokterá aplikace pro Microsoft Surface je v odstínu šedé. Většinou se používají výrazné barvy a techniky kontrastu, kterým se snaží běžné aplikace vyhýbat. Hlavní podíl na tom má fakt, že aplikace bude promítána na skleněnou desku. V případě použité šedé palety, která je obvyklá u drtivé většiny běžných aplikací, by se mohlo stát, že uživatel shledá černý text na šedém pozadí špatně čitelným. Obecně platí, že i kontrasty, které v běžných aplikacích působí rušivě vlivem zvýšené svítivosti a na které se uživatel nevydrží dlouho dívat, se dají v prostředí stolku Microsoft Surface použít.

Posledním prvkem, který jistě uživatele zarazí, je nedostatek jakékoliv nápovědy. Vývojáři aplikací pro Microsoft Surface často spoléhají na to, že ovládání aplikace je dostatečně srozumitelné, aby ho uživatel po chvíli zkoumání pochopil. Způsob realizace nápovědy je v těchto aplikacích poměrně komplikovaný, proto se není čemu divit, že vývojáři od ní často upustí. Běžná nápověda k aplikacím se v prostředí stolku nedá použít, protože vyžaduje používání klávesnice a myši, tedy zařízení, kterými platforma nedisponuje. Nápověda ve formě objektů s textem by zase působily rušivě. Když už se vývojáři rozhodnou použít nápovědu, tak vždy ve formě rad, které po čase zmizí a do nového spuštění aplikace se již neobjeví.

3. Základ pro tvorbu aplikací v Microsoft Surface

Před započetím vývoje aplikace pro Microsoft Surface je třeba si uvědomit základní rozdíl mezi touto platformou a běžnými webovými aplikacemi. Microsoft Surface nemá na rozdíl od běžných aplikací k dispozici klávesnici a myš. Celou aplikaci tak lze ovládat pouze dotekem, což vyžaduje trochu odlišný přístup při tvorbě uživatelského rozhraní. Společná základ ve WPF napovídá v určitých oblastech, jakým způsobem by se měla aplikace chovat a jak vypadat. Jsou zde však jisté rozdíly vyplývající z odlišných vlastností a specifik zařízení.

Protože si aplikace Microsoft Surface zakládají na tom, že práce s nimi budí v člověku pocit reality, je dobré se držet pár pravidel. Při správném pochopení těchto zásad bude vývojář schopen vytvářet aplikace, které budou v uživateli budit dojem, že skutečně pracuje s daty, které aplikace zobrazuje. Osvojení těchto pravidel však vyžaduje čas. Vývojář, který bude s Microsoft Surface zacházet delší dobu, už bude mít tyto postupy zažité a bude schopen odvozovat další. V rámci této kapitoly se uvedou některá pravidla, kterých se vývojáři aplikací pro Microsoft Surface drží a které by mohly pomoci začátečníkům překlenout rozdíly vyplývající z použití jiné platformy než web. Mnohé z těchto rad jsou k nahlédnutí v dokumentaci k Microsoft Surface a jsou ověřeny v praxi [6].

3.1. Plynulá změna

Aby aplikace budila v uživateli pocit reálna, je potřeba zajistit, aby veškerá činnost aplikace probíhala plynule. V reálném světě se jen málo věcí mění skokově, většinou se stav mění postupně, například běžec postupně zpomaluje, než se úplně zastaví. Proto aplikace působí reálnějším dojmem, pokud plynule mění svůj stav, ať už se jedná o změnu pozice, změnu barvy či změnu jiné vlastnosti. Aplikace pro Microsoft Surface mají v tomto výhodu, protože mnohé změny vycházejí z pohybu uživatele, které jsou vždy plynulé. O plynulosti aplikace však nerozhoduje pouze pohyb, proto jsou v následující kapitole popsány některé postupy, které výsledný efekt ještě víc umocňují.

Základní věc, kterou si musí vývojář uvědomit, je, že už nepracuje s běžnou webovou aplikací. Může se to zdát být jako zbytečné konstatování, ale webový vývojáři se určitých konceptů velmi těžko zbavují. A pro ně je směřována první rada. Je třeba zapomenout na diskrétní akce a plně využívat možností Microsoft Surface pro plynulou interakci. Pod pojmem diskrétní akce se skrývá běžný postup zažitý v ovládání aplikací. V drtivé většině se jedná o série úkonů, které mají za úkol provést určitou činnost a které se provádějí zpravidla jedním kliknutím. Příkladem budiž změna rotace, která se často provádějí tak, že se klikne na objekt, kterým se chce rotovat, z nabídky se vybere možnost rotace, tím se objeví v objektu prvky, které to umožní, a následným kliknutím a potáhnutím se objekt otočí podle potřeby. Kliknutím mimo objekt se pak objekt odznačí.

Přitom každý, kdo kdy viděl aspoň jedno prezentační video o Microsoft Surface, si jistě všiml, že žádná zobrazovaná informace se tak složitým způsobem neovládá. Uživatel má volbu si téměř vše upravit podle svého, ať už zvětšením, zmenšením, otočením či pouhým přesunutím na pozici, která mu víc vyhovuje. Přitom to všechno provádí jedním plynulým pohybem. Tato funkce nejlépe vynikne při prohlížení fotografií a platforma Microsoft Surface v tomto plně vytváří iluzi, která v uživateli budí dojem, že pracuje s reálnými fotografiemi a ne jenom s jejich obrazy. Takové iluze by bylo těžko dosaženo pomocí diskrétních akcí. Jedno větou řečeno, je třeba si uvědomit, že Microsoft Surface umožňuje provést většinu jednotlivých operací lépe než pomocí metody „klikni sem a klikni tam.“

Pro tvorbu komponent, kterými pak bude možno v rámci aplikace manipulovat, platí několik pravidel. Asi nejdůležitějším je nutnost používat velký okraj. Každá komponenta, kterou bude manipulováno, by měla mít výrazný okraj, na který bude možno sáhnout. Webový vývojář na tuto podmínku často zapomíná, protože má stále zažito výběr myší. Při použití kurzoru se označí určitý pixel, na základě jehož je vybrána komponenta k manipulaci. Kontakt prstem však označí celou skupinu pixelů. Pokud má jádro systému rozeznat, kterou komponentou má vlastně pohnout, je podmínka použití širšího okraje celkem na místě. To, co platí pro okraj, platí i pro rozestup mezi aktivními prvky. Tyto prvky provádějí určité činnosti a je-li jejich velikost příliš malá nebo rozestup mezi nimi příliš nepatrný, uživatel může mít potíže s vybráním správného prvku (bude například tisknout dvě tlačítka současně).

Obrázek 7: Ukázka chybně rozvržené komponenty

Na uvedeném obrázku je zobrazeno standardní tlačítko. To by mělo být schopné rotovat, ale protože okraj, za který by mělo jít tlačítko uchopit, není téměř viditelný, neexistuje možnost, jak kontakt správně umístit. Každý kontakt, který se provede, bude totiž vyhodnocen jako stisknutí tlačítka nebo jako náhodný kontakt (kontakt na pozadí aplikace). Pokud je uživatel zahnán do situace, kdy musí pečlivě umísťovat kontakty, aby provedl určitou činnost, je narušen plynulý tok jeho práce, což nepochybně bude mít negativní vliv na jeho hodnocení aplikace.

O samotnou plynulost pohybu se postará Microsoft Surface, ale i běžné komponenty by měly plynule měnit svůj stav, pokud zachytí kontakt uživatele. Každý kontakt by měl vyvolat okamžitou viditelnou odezvu, i když to bude znamenat, že počítač bude muset obstarávat další činnosti. Platí, že i malá okamžitá odezva udělá víc, než velká uživatelem volaná potom. Pro zvýšení reálnosti se doporučuje, aby odezva vycházela přímo z místa kontaktu. Při použití vlastních odvozenin běžných komponent je nutné mít definované všechny stavy, které běžná komponenta poskytuje. Například zatrhávací rámeček *Checkbox* poskytuje pro jeden kontakt stavy Nezatržený, Stisknutý a Zatržený. Pokud komponenta vynechá jeden z těchto stavů, působí to na uživatele neplynule a skokově. Je dobré se taky vyvarovat použití kontaktů závislých na čase, jako je například operace, kdy uživatel je nucen tlačítko stisknout a držet. Tyto operace efekt plynulosti spíše kazí.

Doporučuje se také využívat vizuálních pomůcek, které uživateli řeknou, kam a jak má na displej stolku sáhnout. V průběhu činnosti uživatele by neměly nastat okamžiky, kdy nebude vědět, jak má pokračovat. Tyto okamžiky ruší plynulost práce a celkový dojem tak upadá. Je také dobré v mezích možností zajistit, aby se stav komponenty dal vrátit. Microsoft Surface totiž snímá všechny kontakty a je mu jedno, jestli se jednalo o chtěný pohyb či jenom o nechtěné opření dlaně o stolek.

Na závěr kapitoly malé upozornění, Microsoft Surface rozpozná prsty, tagy a různě velké kontaktní plochy (bloby), ale nerozpozná, kterým prstem se uživatel obrazovky dotkl. Stejně tak nerozpozná velké kontaktní plochy, jako je dlaň, paže či neoznačený objekt. Vývojář by měl vědět, co se považuje za kontakt a co ne.

3.2. Práce více uživatelů

Microsoft Surface umožňuje svým uživatelům pracovat v týmu na jedné úloze. Tato vlastnost je často vyzdvihována, téměř v každé prezentaci je zmíněna. Co však zmíněno není, je fakt, že úroveň spolupráce ovlivňuje hlavně způsob napsání aplikace. V případě, že vývojář nemá s těmito aplikacemi zkušenosti, snadno se může stát, že míra spolupráce bude vlivem špatného softwaru snížena. V této kapitole je proto rozebráno, na co si vývojář musí dávat pozor, aby jeho aplikace nabízela uživatelům aspoň určitý stupeň spolupráce.

První, co je třeba si uvědomit, je fakt, že i spolupráce může vypadat různě. Záleží jen na vývojáři, aby určil, který způsob bude preferovat. Definice způsobů spolupráce pochází z hlavy autora a každý vývojář může tyto pojmy cítit jinak.

- Přímá spolupráce – uživatelé spolupracují nad stejnou úlohou, přičemž provádějí stejné činnosti. Příkladem může být vyhledávání v menu, kdy každý z uživatelů prochází určité položky a hledá ty, které je zajímají.
- Rozděl a panuj – spolupráce nad stejnou úlohou je rozdělena. Příkladem budiž například vyhledávání položek, kdy jeden uživatel vyhledává názvy kategorií a druhý je zadává do vyhledávače položek, aby je třetí mohl prohlížet.
- Každý sám na sebe – nad společným prostorem si každý uživatel dělá, co chce. Například si jeden hledá kontakty a druhý ve stejné kategorii hledá fotografie.

Vývojář by si hned na začátku vývoje měl uvědomit, který způsob spolupráce bude asi v jeho aplikaci nejvíce využíván a jak velká skupina uživatelů bude mít k aplikaci přístup. Pokud je aplikace určené do rukou soukromé osoby, je vhodné, aby uživatel mohl s aplikací pracovat sám, případně aby aplikace vyžadovala jen minimální počet účastníků. Na druhou stranu aplikace nacházející se na veřejně přístupném místě musí být schopná zvládnout manipulace více uživatelů a umožnit co nejvyšší míru přímé spolupráce.

Pro aplikace, u kterých se obecně předpokládá, že s nimi bude pracovat více uživatelů současně, se doporučuje dodržovat pár pravidel a rad. Nový uživatel by měl mít možnost se přidat, aniž by rušil ostatní uživatele. Stejně tak, pokud jeden uživatel ukončí svou činnost a ostatním uživatelům se práce zavře též, není to dobrý příklad správně uchopené aplikace pro více uživatelů. Je jisté, že mezi uživateli nastane určité dělení pracovního prostoru. Dva uživatele těžko budou ovládat aplikaci ze stejného místa. Nejdůležitější radou určenou pro tyto aplikace je tedy dodržovat odstup mezi účastníky. Příkladem budiž uzamknutí ovládacích prvků k jedné straně, což ve výsledku způsobí, že se uživatel musí naklonit až k dalšímu účastníkovi, aby mohl provést nějakou operaci. Ani pro jednoho by to nebyla příjemná zkušenost. Ze stejného důvodu se nedoporučuje fixovat orientaci ovládacích prvků.

Zvláště u aplikací pro více uživatelů, kde každý pracuje sám na sebe, není dobré, aby komponenty ovlivňovaly své kolegy. Příkladem budiž aplikace malování, kde uživatel vytvoří nové plátno, aby mohl malovat. Pokud aplikace již jedno plátno obsahuje a při vytvoření druhého smaže

obsah prvního, nebude jistě uživatel, který byl u aplikace první, rád. Podobný problém vzniká při používání komponent, které se posouvají po ploše v závislosti jedné na druhé.

Pokud se už dopředu ví, že aplikace bude umožňovat spolupráci více uživatelů nad jednou úlohou, je třeba dodržovat pravidlo, že není dobré fixovat ovládací prvky na určité místo. Důvodem je, že autor aplikace nemůže vědět, z které strany bude uživatel k aplikaci přistupovat. Ze stejného důvodu není dobré dělit prostor Microsoft Surface na oblasti pro jednotlivé úlohy. V mezích možností a znalostí by měl vývojář umožnit uživatelům úlohu paralelizovat. V rámci těchto aplikací by se měl vývojář zcela vyvarovat diskrétních akcí. Každý z uživatelů má právo změnit si prostředí podle svého, ale ostatní účastníci by měli vidět, jak to provádí, například zvětšení okna pomocí dvou prstů roztažených od sebe a ne použitím tlačítka někde v rohu displeje.

Pro multi-dotykové aplikace všeobecně se nedoporučuje využívat zvukových signálů, jako je třeba cinknutí, když uživatel provede chybnou operaci. V prostředí Microsoft Surface, kde lze provádět až 52 kontaktů současně, by se jen těžko určovalo, který z kontaktů vlastně chybu způsobil. Co se týká optimální pozice pro ovládací prvky, pokročilí vývojáři doporučují, aby komponenty, které jsou uživateli vlastní, se nacházely přímo před ním, naproti tomu společné ovládací prvky by měly být v prostředku plochy, aby k nim měli všichni uživatelé stejně daleko.

Na závěr kapitoly malé doporučení, aby většina aplikací byla vytvářena s cílem postavit uživatele proti sobě. Pokud má aplikace specifičtější užití, kdy každý uživatel bude mít přístup k jiným funkcím, je toto dělení plochy praktičtější. Příkladem budiž banka, kdy bankovní úředník stojí na jedné straně stolku a zákazník na druhé. Přidáním stejných funkcí na opačnou stranu stolu se dá jednoduchým způsobem vytvořit aplikace pro více uživatelů, kteří budou pracovat samostatně. Je třeba si však ohlídat, aby se stejné komponenty neovlivňovaly.

3.3. Prostorovost

Většina běžných aplikací se do určité míry snaží zanést do svého uživatelského prostředí prvek prostorovosti. Jedná se o iluzi, kdy se uživateli zdá, že pracuje v 3-D prostoru. V rámci obyčejných, plochých aplikací se dá tento prvek simulovat jen vizuálně. Jednotlivé objekty mají například stín, zešikmené hrany (které budí v uživateli dojem, že se na objekt dívá z jiného úhlu, než přímého) nebo se navzájem překrývají. Běžně se říká, že aplikace ovládá 2.5-D prostor.

Obrázek 8: 3-D v Microsoft Surface

Pro reálný dojem z práce s daty je tato vlastnost klíčová, a proto se ji snaží Microsoft Surface plně podporovat. I začátečníci jsou tak tuto podmínku schopni plnit, protože Microsoft Surface udělá část práce za ně a jak je z obrázku patrné, každý objekt, kterým se dá manipulovat, má po uchopení svůj stín a budí zdání, že se nadzvedl z povrchu.

Jelikož za 3-D chování se považuje i možnost objekty zvětšovat, zmenšovat a různě je natáčet, je mnoho rad z předcházejících kapitol platných i zde. Radou navíc je, že v aplikacích Microsoft Surface nejde používat typické pojetí projekce prostoru, jako je například vlevo, vpravo, nahoře a dole. Jednoduchou funkcí lze totiž orientaci celé aplikace obrátit, takže nápovědy typu „tlačítko vlevo dole“ pozbývají smyslu. Začínající vývojáři také často zapomínají na to, že skutečné rozměry stolku se podstatně liší od náhledu, který mu poskytuje počítač během vývoje aplikace. To, co vývojáři připadá snadno dosažitelné, může vypadat z pohledu uživatele stolku jinak.

Realitu aplikace hodně ovlivňuje vzhled komponent. Při výrobě vlastních komponent je tedy dobré určitým způsobem naznačit zdání prostorovosti, například obyčejným stínem. Takové komponenty přitahují uživatelovu pozornost a jejich funkce se pak mnohem lépe učí. Vhodné je také nevnucovat prvkům přesnou pozici a orientaci, protože nelineární a trochu chaotické rozprostření komponent po ploše přitahuje pozornost uživatele, který se bude snažit dát těmto prvkům řád, který víc odpovídá jeho požadavkům. Díky tomu se s ovládáním seznámí lépe než by se naučil z manuálu. Autor aplikace by měl také zajistit, aby texty byly čitelné a ovládací prvky dávaly smysl ze všech stran stolku. Například tlačítko, na kterém je šipka ukazující vlivem volné orientace jakýmkoliv směrem, může uživatele jenom zmást. Rada na závěr, vývojář by si měl uvědomit skutečné rozměry stolku. Když bude mít tuto skutečnost stále na paměti, nemůže se v jeho aplikaci stát, aby uživatel musel obcházet stolek, aby se dostal k funkci, kterou zrovna potřebuje.

3.4. Přirozenost

Jednou z hlavních vlastností, na které si Microsoft Surface zakládá, je schopnost vytvořit iluzi reálné manipulace s prostředím. Ačkoliv některé prvky, se kterými uživatel pracuje, někdy nefungují na reálném základě, například fotografie se ve skutečnosti dají jen těžko zvětšit, manipulace s nimi je intuitivní a vychází z reálných pohybů (kdyby se fotografie daly zvětšovat, jistě by se to provádělo roztažením hran, jako je to v případě Microsoft Surface). Díky tomu se veškeré manipulace provádějí přirozeně a uživatel si je velmi rychle osvojí.

V rámci tohoto textu již byla přirozenost neboli pocit reálné manipulace s daty několikrát zmíněna. Hodně udělá reálný vzhled komponent, plynulost jejich akcí a okamžitá reakce na kontakt, která zatáhne uživatele do hloubky aplikace a efekt reálnosti je mnohem intenzivnější. Jsou však i další možnosti, jak přirozenost v rámci aplikace podpořit.

Pro menu a různé seznamy, větších než je zobrazovací okno, platí nutnost používat operaci scrollování, tedy potáhnutí obsahu do výřezu okna, aby se zobrazilo to, co je zatím schované. Pokud autor aplikace chce, aby uživatelé měli pocit reálna, měl by se použití této operace pokud možno úplně vyhnout. V prostředí Microsoft Surface je možno, v případě, že obsah menu je nutné posunout, použít operace šťouchnutí, kdy jemným pohybem prstu uživatel menu uvede do pohybu a to se posune o kus dál. Tato operace se dá použít i při pohybu komponent po ploše, kdy při naražení na hranu displeje se komponenta odrazí zpátky. Dojem z takového způsobu scrollování je mnohem přirozenější než v běžném případě.

Pro zvýšení zážitku uživatele se doporučuje zavést do aplikace určitou míru života. Jedná se o efekty, jaké jsou například v aplikaci typu vodní hladina, kdy se občas po hladině rozeběhnou kruhy, což působí mnohem lepším dojmem, než kdyby byla aplikace úplně statická a čekala na kontakt uživatele. Je třeba si však dávat pozor na to, aby tyto změny nepůsobily rušivě a rozptylujícím dojmem. Nedoporučuje se měnit stav hlavních komponent, tyto občasné změny by se vždy měly týkat jen pozadí, případně grafického vzhledu komponent.

3.5. Tagy

Microsoft Surface disponuje i unikátní technologií rozpoznávání objektů. Díky tomu lze do aplikací zakomponovat i objekty, které se položí na stolek. Příkladem budiž již velmi známé aplikace typu galerie fotografií, které po přiložení mobilního telefonu na povrch stolu rozpoznají, jaké fotografie se nacházejí v jeho paměti. Za touto technologií se ukrývá tag. Pod tímto názvem se skrývá speciální vzor bodů, který obsahuje různé geometrické v infračerveném světle rozpoznatelné objekty. Microsoft Surface rozpoznává dva typy tagů: Byte tagy (taky pojmenováno jako Domino tagy) a Identity tagy.

Mezi oběma typy jsou markantní rozdíly, jejich funkce je však vždy stejná: rozpoznat předem definovanou hodnotu a tím i daný předmět. Tagy jednotlivých typů se dají snadno charakterizovat a rozpoznat [7].

Byte Tagy

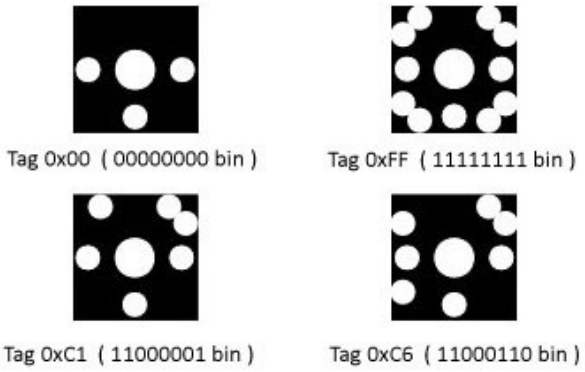
- Hodnota se ukládá na 8 bitů (1 byte)
- 256 jedinečných hodnot
- Menší rozměr (3/4 x 3/4 palce)
- Spolehlivé rozpoznávání i při pohybu tagu
- Atribut *ByteTag.Value* reprezentuje hodnotu tagu

Identity Tagy

- Hodnota se ukládá na 128 bitů (dvě 64 bitové hodnoty)
- Mnohem větší rozsah jedinečných hodnot (přibližně 340,282,366,920,938,000,000,000,000,000,000,000,000)
- Větší rozměr (1 x 1 palce)
- Spolehlivější pro tagy, se kterými se nemanipuluje
- Hodnota tagu je reprezentována dvěma atributy (*IdentityTag.Series* a *IdentityTag.Value*), které společně tvoří jednoznačný identifikátor tagu

Zásadní nevýhodou tagů je nutnost používat specifických materiálů. Běžné tiskařské barvy totiž neodrážejí infračervenou složku světla. Ze stejného důvodu nelze použít obyčejný papír, u kterého navíc hrozí nebezpečí rozmazání tagů. Pro rozpoznání musí být tag zřetelný, proto se používá vinylový papír s rychlou dobou zasychání. Náklady na tisk tagů jsou tak vyšší než náklady na tisk obyčejných dokumentů. Určitým problémem také je, že málokterý uživatel zpočátku ví, jak hodnoty jednotlivých tagů vypadají.

Tagy se v běžné praxi nepoužívají, pro identifikaci objektů se běžně používá jiných kódů (například čárový kód). Firma Microsoft tento problém vyřešila tím, že pro tvorbu Byte tagů vytvořila šablony, ve kterých je ukázáno, jak která hodnota vypadá (přece jen jich je pouze 256). Tyto materiály jsou volně ke stažení [8].

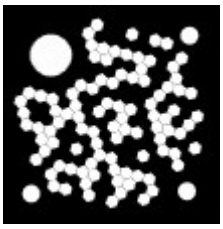


Obrázek 9: Ukázka Byte tagů

Kvůli množství hodnot se pro tvorbu Identity tagů šablony nepoužívají. Místo toho se využívá nástroje Identity Tag Printing Tool, který se nachází v základní nabídce nástrojů pro Microsoft Surface. Vzhledem k tomu, že cena zařízení Microsoft Surface je tak vysoká a nový vývojáři přicházejí jen poskrovnu, je tento nástroj zatím jediná možnost, jak tisknout Identity tagy. Nejspíš také díky tomu se ve všech prezentacích používají pouze Byte tagy a Identity tagy se zatlačují do pozadí.

Ať už autor aplikace zvolí jakýkoliv typ tagů, měl by se držet několika pravidel. Každý tagovaný objekt položený na povrch stolu by měl mít okamžitou odezvu. Nejen proto, aby uživatel objekt opět neodstranil, ale i proto, že okamžitá reakce stírá rozdíl mezi reálným a nereálným. A aby práce s objekty na stolku připadala uživateli reálná, o to by vývojáři mělo jít především. Než vývojář začne navrhovat vzhled tagu, musí si uvědomit vzhled a rozměry objektu, který bude tímto tagem označen. V základním nastavení je totiž tag zobrazen na místo kontaktu. Může se tak stát, že objekt zakryje zobrazený tag a uživatel je ochuzen o další funkce, které tag skýtá.

Při použití objektů, jež se mají s počítačem Surface propojit, je nutné si uvědomit, že navázání spojení vyžaduje určitou dobu. Proto je dobré uživatele upozornit, že se o daném objektu ví. Třeba i obyčejné kolečko, točící se kolem mobilního telefonu položeného na stolku, uživatele upozorní, že se něco děje. Je vhodné použití tagů dělit. Pro mobilní telefon Siemens A50 se použije Byte tag, pro mobilní telefon Siemens A50 s určitým telefonním číslem se použije Identity tag. Tím se zamezí tomu, aby se možné hodnoty Byte tagu vyčerpaly.



Obrázek 10: Ukázka Identity tagu

Objekty, které se v rámci aplikace jeví jako jedinečné, například zmiňovaný mobilní telefon s určitým telefonním číslem, by měly být určitým způsobem odlišeny od běžných objektů. Například se dá použít upravené animace, chovající se pro specifické objekty jinak než obvykle, čímž upozorní uživatele na fakt, že objekt je něčím význačný.

Poslední věc, na kterou pokročilí vývojáři upozorňují, je odezva objektů, jež odrážejí infračervené světlo. Mohou to být mobilní telefony s lesklým lakem nebo jiné běžné objekty. Tyto objekty zbytečně zatěžují rozpoznávací jednotku, která v takovém objektu hledá hodnotu tagu, kterou pochopitelně nenajde. V některých případech může takový předmět způsobit, že tag položený v jeho blízkosti nebude rozpoznán nebo celá aplikace bude zpomalená přílišným zatížením jádra platformy. Pokud tedy uživatel vidí, že tag v kombinaci s daným předmětem nemá žádnou odezvu nebo způsobuje problémy spojené s činností aplikace, měl by se pro příště jeho použití vyvarovat.

3.6. Estetika

Estetika je kámen úrazů mnohých začínajících vývojářů. Aplikace po funkční stránce dokonalé, nenajdou mezi uživateli své uplatnění, protože jsou nepřehledné a mají špatně zvolený vzhled. Toto jediné se vývojář nemůže naučit, chce to léta praxe a cit pro grafiku. Jediné, co lze pro tyto jedince udělat, je upozornit na jejich chyby a vysvětlit jim, čeho by se měli příště vyvarovat. A to je obsahem této kapitoly.

Vzhled aplikace by měl brát v potaz skutečnost, jakým je aplikace prezentována. V případě, že aplikace se má zobrazovat na monitorech, je třeba si uvědomit, že vyzařování obrazovky může působit na uživatele rušivě. Proto se nedoporučuje používat výraznou barvu pozadí, z toho důvodu, že uživatel se na takovou aplikaci nevydrží dívat dlouho. Microsoft Surface je v tomto ohledu výjimkou, protože způsob promítání umožňuje použít i výraznějších barev. Přesto se však doporučuje použít barev mlhavého odstínu, které na lidské oko nepůsobí rušivě. Například jasně červené pozadí není případ dobře zvolené barevné palety. Fakt, že obraz je promítán na skleněnou desku, by měl také napovědět, že využívat jemných odstínů pozbývá smyslu. Komponenty, které využívají podobných barevných motivů s pozadím, nemusí být snadno rozpoznatelné. Stejné pravidlo, dodržovat kontrast, platí i pro text. Pro volbu barevné palety platí, že by měla být pro všechny komponenty totožná. Jedna komponenta s červeným pozadím a jiná se zeleným působí jen rušivým dojmem.

Aby byl zážitek uživatele co největší, je doporučeno se zaměřit na víc než jeden smysl. Lze využít více aspektů Microsoft Surface současně, jako je například vzhled, zvuk, pohyb a fyzická interakce. Pro použití zvuků v tomto prostředí platí jedno jednoduché pravidlo. Zvuky se dají využít jen u operací, které se často neopakují. Příkladem budiž již v předchozí kapitole zmíněný vstupní operace, kterých uživatel může udělat tisíce. Zde by zvuk, upozorňující na to, že se uživatel spletl a sáhl vedle, působil jen rušivě.

3.7. Desatero začátečníka

Rad a návodů, jak má správně aplikace pro Microsoft Surface vypadat, je mnoho a není v moci začínajícího vývojáře si všechno zapamatovat. Většinu z těchto znalostí získá postupem času sám, jak bude vyvíjet další a další aplikace. Některé základy je však dobré mít na paměti již od začátku, aby případná aplikace poskytovala aspoň minimum vlastností, jimiž se Microsoft Surface honosí. K tomuto účelu vytvořil autor tohoto textu malé shrnutí nejdůležitějších rad, kterých se držel při tvorbě svých aplikací pro Microsoft Surface.

1. **Nenechat se omezit komponentami Surface.** Microsoft Surface obsahuje jen nejnutnější komponenty pro tvorbu aplikací. Většina z nich má vágní vzhled a není jich tolik, aby uživatelské prostředí bylo bohaté. Proto se vývojář nesmí bát využívat obyčejných komponent z běžné nabídky nebo vytvářet své vlastní, vzhledově zajímavější typy.
2. **Výrazný okraj pro manipulaci s komponentami.** Důležitá rada, aby aplikace byla plně funkční. Vývojář by měl stále mít na paměti, že kontakt prstem není to samé, co kontakt myší. Tam, kde myši stačí jeden pixel, jich prst potřebuje celou skupinu.
3. **Každá komponenta má mít svou odezvu.** Aby aplikace vypadala reálně, musí každá položka v aplikaci reagovat na vnější stimul (dotek prstem, položení tagu). Komponenta, která jenom provede svou činnost, a ani trochu se nezmění, je komponenta, která ničí dojem reálnosti.
4. **Monitor není stolek.** Vývoj aplikace se většinou provádí na stolním počítači, ale pro něj aplikace určena není. Vývojář by měl mít na paměti, že ačkoliv může aplikace na monitoru působit estetickým dojmem, na stolku bude vypadat docela jinak.
5. **Uvědomit si skutečné rozměry stolku.** Tato rada souvisí s předcházející. To, co na monitoru je zobrazeno nahoře, je v prostředí stolku většinou zobrazeno na druhé straně. Pokud vývojář nechce, aby uživatel musel obcházet celý stůl, měl by přizpůsobit pozici komponent podle toho. Stejně tak je třeba mít stále na paměti, že v případě, že aplikace je určena pro více uživatelů současně, těžko můžou dva uživatelé používat komponenty z jednoho místa.
6. **Přirozenost, přirozenost, přirozenost.** Vývojář by měl mít stále představu o tom, jak by se s prostředím aplikace pracovalo, kdyby byla skutečná, například ve formě stolní desky. Co by se dalo posouvat, co otáčet, jakým způsobem by se objekty zvětšovaly, to všechno musí vycházet z přirozených pohybů. Pokud se tyto přípravy provedou pořádně, je aplikace přirozená a uživatel bude mít pocit, že pracuje s realitou.
7. **Malá reakce ihned je lepší než velká animace po chvíli.** Důležité pravidlo, hlavně pro používání tagů, hodí se však i při vývoji vlastních komponent. Pokud se uživatel dotkne komponenty, čeká, že uvidí nějakou reakci. Pokud na sebe nechá reakce dlouho čekat, působí to rušivým dojmem. Netřeba všude používat animace, i obyčejná změna barvy někdy způsobí víc.
8. **Barvy mohou být zrádné.** Projektor v základně stolku způsobuje, že jemnější odstíny se vytrácí. Hnědý text na žlutém pozadí bude tedy stěží čitelný. Proto opatrně na volbu barev.
9. **Tagy používat s mírou.** Aplikace předpokládající vstup ve formě tagů je k ničemu, pokud uživatel nemá tag zrovna k dispozici. Příliš velké množství tagů bude působit rušivě a málokdo pak bude mít o takovou aplikaci zájem.
10. **Nebát se experimentovat.** Nejdůležitější pravidlo na závěr. Microsoft Surface nabízí mnoho možností, jak ozvláštnit běžné aplikace. Množství funkcí, které je k mání, se hned tak neokouká. Je třeba toho využít a experimentovat, jak je do svých aplikací zavést a jakým přínosem budou. A hlavně, vývojáře to musí bavit.

Tím končí tato kapitola. Mnohé rady, které se zde vyskytují, se týkají i běžných webových aplikací. Netřeba se bát, že některé podmínky nejde splnit, Microsoft Surface těmto podmínkám všemožně vychází vstříc a snaží se, aby i pro začínajícího vývojáře nebylo těžké je plnit.

4. Praktické postupy pro vývoj aplikací v Microsoft Surface

Až dosud byly všechny informace týkající se vývoje aplikací pro Microsoft Surface v teoretické rovině. Bylo zmíněno, co pro vývoj aplikace použít, na co si dávat pozor a čeho se při procesu vývoje aplikací vyvarovat. Zatím však nebylo vysvětleno, jakým způsobem dosáhnout určitého efektu. K tomu je určena tato kapitola, která si klade za cíl dát začínajícím vývojářům přehled o základech implementace Microsoft Surface.

Vývoj aplikací pro Microsoft Surface se moc neliší od vývoje běžných webových aplikací. Společný základ tvoří Windows Presentation Foundation (WPF) a technologie Windows XNA, díky čemuž se pokročilý webový vývojář nebude muset všechno učit znovu. Výhodu budou mít ti, kteří pracují s podobnými technologiemi, jako je například Microsoft Silverlight.

4.1. Nástroje pro tvorbu aplikací

Samotná instalace prostředků pro vývoj Microsoft Surface aplikací je jednoduchá záležitost. Instalační balíček sám zjistí, zda bude prostředí počítače vyhovující. Pokud shledá rozpory s potřebami Microsoft Surface, upozorní na to, co se musí změnit. Po instalaci je vše připraveno na používání, ne třeba provádět žádné registrace a změny nastavení. Samotná aplikace se dá vyvíjet v jakémkoliv běžném editoru, doporučuje se však používat Microsoft Visual Studio 2008 či jemu podobný.

Všeobecně pro tvorbu jakékoliv aplikace na bázi WPF se doporučuje, aby autor využíval možností Microsoft Expression Blend. Tento nástroj je speciálně určen pro tvorbu vzhledu aplikací a nabízí v tomto směru mnohem víc možností než Visual Studio. Pokud uživatel vlastní oba tyto nástroje, nabízí se mnohem efektivnější způsob vyvíjení aplikace. Visual Studio je lepší používat pro funkční kostru aplikace (lépe manipuluje s proměnnýma, mnohem lepší nápověda a v neposlední řadě možnost odlaďování), zatímco Microsoft Expression Blend se dá použít pro ladění vzhledu aplikace (má mnohem lepší možnosti vizualizace, funkce pro tvorbu vlastních tvarů, přehledné nabídky vlastností komponenty a jiné). Je to však pouze návrh, k samotnému vývoji aplikace pro platformu Microsoft Surface stačí libovolný z těchto nástrojů.

Mimo softwarové balíčky a knihovny uživatel instalace získá i přístup k několika nástrojům, které mu mohou usnadnit práci. Tím nejdůležitějším je Microsoft Surface Simulator. Jak je patrné z názvu, tento nástroj umožňuje na běžném stolním počítači simulovat chování stolku Microsoft Surface. Funkce doteku je zde nahrazena myší a sérii kurzorů, které simulují jednotlivé typy kontaktů (prst, blob, tag). Simulator umožňuje použít i více myší současně, takže lze zkoušet i složitější manipulace.

4.2. Komponenty Microsoft Surface

Microsoft Surface umožňuje použít mnohé z běžných komponent. Použití některých je však omezeno, zvláště u těch, které vyžadují aktivní přispění uživatelé, například tlačítko, které čeká na kliknutí myší. Protože nelze použít jiného kontaktu než doteku, tyto komponenty by se korektně zobrazily, neměly by však žádnou funkci. Naštěstí je Microsoft Surface vybaven sadou vlastních komponent, které plní funkci těch běžných. Ve většině případů mají i shodný název, který koresponduje s původním, například *Button* a *SurfaceButton*. Mnohé z komponent Microsoft Surface tedy mají své protějšky v běžných aplikacích. Tento text se však zaměřuje hlavně na ty, které se vyskytují pouze v prostředí Microsoft Surface. Nejčastěji používaná komponenta vůbec, patřící do této kategorie, je bezesporu *ScatterView*.

Tento kontejner má podobnou vlastnost jako například *Grid.* Jedná se o objekt na pozadí, jež uživatel přímo nevnímá, ale jeho nepřítomnost by nepřímo ovlivňovala funkci či vzhled aplikace. Zatímco *Grid* má za úkol ukotvit podřízené prvky v určité pozici, *ScatterView* má naopak zajistit, aby pozice objektů byla čistě náhodná a uživatelem změnitelná. Díky tomu každá položka umístěna do kontejneru *ScatterView* získá schopnost být v prostředí Microsoft Surface posouvána, rotována, zvětšována i zmenšována pomocí kontaktů, tedy schopnost, kterou uživatelé oceňují nejvíce.

Obrázek 11: Objekt *Label* ve *ScatterView*

Položkou kontejneru *ScatterView* může být cokoliv. Lze vkládat tlačítka, textová pole či uživatelem vytvořené komponenty. Schopnost manipulace pomocí kontaktů získají díky tomu, že budou přiřazeny k objektu *ScatterViewItem*, které plní funkci položek kontejneru. Při vložení do *ScatterView* dojde automaticky k umístění objektu jako potomka *ScatterViewItem,* kterým se dá v prostředí Surface manipulovat. Díky tomu lze pak s objektem manipulovat, ačkoliv uživatel nemanipuluje přímo s komponentou, ale s objektem *ScatterViewItem,* na kterém je komponenta umístěna.

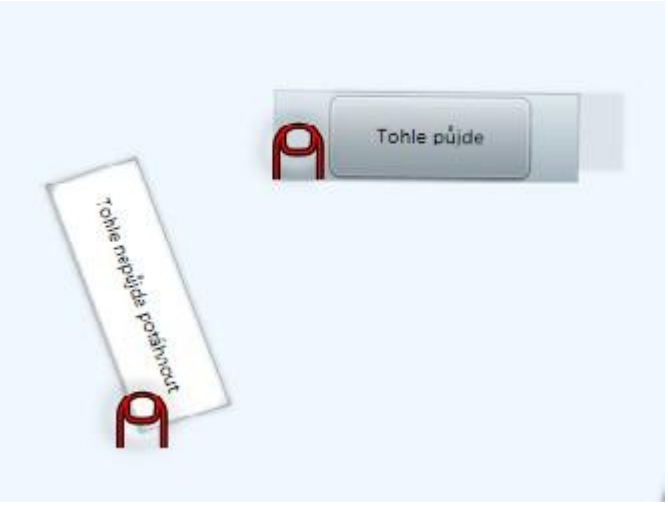
Položku *ScatterViewItem* lze i přímo definovat. Ačkoliv se tato položka definuje automaticky pro každý nový objekt v kontejneru *ScatterView,* v některých případech je dobré přímo položku definovat a pak přiřadit její obsah. Účelem tohoto postupu je získat možnost manipulovat s vlastnostmi *ScatterViewItem.*

```
<s:ScatterView Background="{StaticResource WindowBackground}">
      <s:ScatterViewItem CanRotate="False">
            <Label>Nelze rotovat</Label>
      </s:ScatterViewItem>
      <Label>Lze rotovat</Label>
</s:ScatterView>
```

Obrázek 12: Zdrojový kód definování *ScatterView*

Tato ukázka kódu definuje kontejner *ScatterView* a v něm dvě položky typu *Label.* Rozdíl mezi nimi je, že jednou je schopen uživatel otáčet a druhou ne. Na uvedeném příkladu je také vidět, jakým způsobem se dá ovlivnit manipulace s objekty. Lze například zakázat rotaci, určit přesnou pozici, na které se bude objekt objevovat nebo zadat objektu jednoznačný identifikátor, pomocí nějž pak bude objekt v kontejneru snáze vyhledáván.

Na vztahu *ScatterViewItem* a jeho podřízené položky se dá vysvětlit pravidlo, proč objekt musí mít široký okraj. Objekty v pozadí, jako je *Grid,* se v prostředí *ScatterView* stanou položkami *ScatterViewItem.* Tím se stane objekt schopným manipulace. Pokud však většinu prostoru vyplňuje aktivní prvek, například tlačítko, nebude fyzicky možné na *ScatterViewItem* sáhnout. Jakýkoliv kontakt se totiž přečte jako kliknutí na tlačítko nebo sáhnutí do prostoru.



Obrázek 13: Správný a špatný příklad objektů ve *ScatterView*

Na obrázku je vidět, že zatímco první kontakt zvedne tlačítko z povrchu a je možno ho přesunout jinam, v případě druhém je okraj *ScatterViewItem* téměř neznatelný a jakýkoliv kontakt se přečte jako stisknutí tlačítka.

Až do nedávna bylo použití *ScatterViewItem* spojeno s určitým problémem, který byl vývojář nucen řešit. Tímto problémem byla omezená počáteční velikost položek, které se ukládaly do kontejneru *ScatterView*. Toto omezení zabraňovalo tomu, aby při inicializaci aplikace byly položky kontejneru větší než 20% celkové velikosti kontejneru a způsobovalo, že se komponenty, které vývojář vytvořil, po umístění do kontejneru *ScatterView* nezobrazovaly korektně. Často se stávalo, že se v kontejneru objevila jen polovina komponenty, protože *ScatterView* upravil její rozměry tak, aby splňovala podmínku o počáteční velikosti. Položku sice šlo hned zvětšit a celou komponentu tak odhalit, ale první dojem tak trpěl. Tento problém dlouhou dobu znamenal, že autor aplikace musel upravovat komponenty tak, aby byly menší nebo aby neodhalení celé komponenty nepůsobilo nedodělaným dojmem. Nyní už tyto metody postrádají smyslu, protože service pack pro Microsoft Surface tento problém opravuje.

4.3. Problémy vyplývající z nedostatku jiných zařízení

Důraz této kapitoly je kladen na komponenty a objekty, které svým složením vyžadují přístup k některému vstupnímu zařízení. Tyto komponenty mají ze zřejmých důvodů v prostředí Microsoft Surface vážné problémy, protože kontakt je jediná událost, která má vliv na stav komponenty. Není problém je zobrazovat, ale je problém s nimi pracovat. *TextBox* sice zobrazí okno se základním textem, uživatel však do něj nemá možnost psát (chybí klávesnice). *Canvas* zase neumožňuje kreslení, protože očekává, až uživatel použije levé tlačítko myši (která tam ale není). Microsoft Surface sice umožňuje v omezené míře připojovat vnější zařízení, ale aplikace by na ně neměly spoléhat. Vývojář

by si měl uvědomit, že spoléhání na jiné vstupní zařízení může vést až k vytěsnění zážitku z přímé interakce s daty.

Problém s nepřítomností vnějšího ovládacího zařízení se dá řešit hned dvojím způsobem. První způsob je použití komponent, které vycházejí z těchto problematických objektů a jsou uzpůsobeny pro použití v Microsoft Surface. Tyto komponenty, až na drobné výjimky, začínají vždy slovem *Surface*. Svým vzhledem se moc neliší od svých běžných protějšků. Obsahují však funkce potřebné pro provoz na stolku Surface. Příkladem budiž komponenta *SurfaceTextBox,* která, když rozpozná kontakt, zobrazí klávesnici, kterou lze opět pomocí kontaktů používat.

Obrázek 14: *SurfaceTextBox* a příslušná klávesnice

Druhým řešením je využít standardních komponent, při nichž se ale využije funkcí z třídy *Contacts*. V této třídě se nacházejí události, které svým způsobem korespondují s funkcemi běžné myši. Příkladem budiž událost *ContactDown* a její protějšek *MouseLeftButtonDown*. Tento způsob řešení je trochu komplikovanější a nikdy se nedosáhne takové svobody pohybu, jako při použití komponent Microsoft Surface určených ke stejným účelům, přesto však své uplatnění nacházejí.

```
<Button x:Name="Klik" s:Contacts.ContactDown="Klik_ContactDown">
      Funguju v prostredi Microsoft Surface
</Button>
```

Obrázek 15: Přidělení události *Contacts* běžné komponentě

Tlačítko, které by v příkladu bylo zobrazeno, by po kontaktu sice nezměnilo svůj vzhled (nebylo by vidět, že tlačítko je stisknuté), ale událost by se při kontaktu stejně zavolala. Komponenta *Button* má navíc možnost použít událost *Click,* která má v prostředí Microsoft Surface výjimku a zachází se s ní jako s kontaktem.

4.4. Používání tagů

Co to je tag a co všechno umožňuje, bylo řečeno již v předcházející kapitole. Zatím však nebylo vysvětleno, co se za schopností rozpoznávat tagy skrývá. Není to nikterak složité, využije se funkcí komponenty zvané *TagVisualizer*. Tato komponenta se na první pohled chová jako *ScatterView*. Je umístěna v pozadí aplikace a na první pohled není patrná. Pečlivě však skenuje každý kontakt na povrchu stolku a srovnává ho s množinou tagů, které má v paměti. Pokud některý vzor odpovídá, dojde k spuštění dalších akcí. Obvykle se jedná o zobrazení vizualizace tagu.

Rozdíl mezi komponentami *ScatterView* a *TagVisualizer* spočívá v tom, že *TagVisualizer* potřebuje předem určit, které hodnoty má hledat. V případě Byte tagů je možných hodnot 256, v případě Identity tagů několik miliard. Takové množství dat se v paměti udržet nedá, a proto musí vývojář rozhodnout, které hodnoty tagů budou v jeho aplikaci aktivní. Tag, který není definovaný, nevyvolá žádnou reakci. Pokud má tagovaný objekt umístěný na povrch stolku platnou hodnotu, snímací systém rozpozná tag a určí jeho hodnotu, pozici a orientaci. Na základě hodnoty tagu pak zavolá vizualizovanou podobu tagu.

```
<s:TagVisualizer>
      <s:TagVisualizer.Definitions>
            <s:ByteTagVisualizationDefinition
                  Value="195" Source="Tag.xaml">
            </s:ByteTagVisualizationDefinition>
      </s:TagVisualizer.Definitions>
</s:TagVisualizer>
```

Obrázek 16: Definování tagu ve WPF

Příklad ukazuje způsob, jakým se tag definuje. Takto zadaný *TagVisualizer* rozpozná Byte tag s hodnotou *Value* 195 (0xC3). Pokud takový tag bude rozpoznán, zavolá se panel, která se skrývá v souboru, zadaném atributem *Source*. Dva jmenované atributy, *Source* a *Value* jsou povinné, bez nich nepůjde tag zobrazit. Mimo tyto atributy lze definovat i další pojmy, jako vektor posunutí objektu, aby se panel neobjevoval přímo na místě kontaktu, ale například o něco níž. Dále lze uvést počet kontaktů s daným tagem, které může *TagVisualizer* přijmout, orientaci zobrazeného tagu a další.

Podobně jako všechny komponenty v prostředí Windows Presentation Foundation lze i *TagVisualizer* zadávat programově.

```
TagVisualizer visual = new TagVisualizer();
ByteTagVisualizationDefinition define
      = new ByteTagVisualizationDefinition();
define.Source = "Tag.xaml";
define.Value = 195;
visual.Definitions.Add(define);
```

Obrázek 17: Definování tagu programově

Pokud je vše správně nadefinováno, měl by být *TagVisualizer* schopen rozpoznat položený tag a zavolat komponentu na definované adrese. Pro tuto komponentu však také platí jedno důležité pravidlo. Obsahem komponenty může být cokoliv, ale celý soubor musí být typu *TagVisualization.* Kontejner *TagVisualizer* není schopen zobrazovat okna, uživatelem definované komponenty, stránky ani jiné komponenty definované v záhlaví souboru. Pokud se tedy tag nezobrazí, ačkoliv by měl, vývojář by měl zkontrolovat právě tuto podmínku, často se na ní zapomíná.

Uvedený kód představuje úplnou definici vzhledu tagu. V záhlaví kódu je komponenta určena jako *TagVisualization,* takže pokud bude přiřazena k definovanému tagu, bude korektně zobrazena. Pokud by v záhlaví bylo napsáno například *Page* (tedy že celý soubor popisuje webovou stránku), komponenta *TagVisualizer* by sice soubor přečetla, ale nebyla by schopna ho zobrazit.

```
<s:TagVisualization
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:s="http://schemas.microsoft.com/surface/2008"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"
x:Name="tagVisualization"
Width="380" Height="380">
<Grid x:Name="LayoutRoot" Margin="0" Width="381" Height="381">
      <Ellipse x:Name="ellipse1" Stroke="#FFDEDEDE" Width="380"
      Fill="#FFDEDEDE" d:LayoutOverrides="HorizontalAlignment"
      HorizontalAlignment="Left" Height="380"
      VerticalAlignment="Top" RenderTransformOrigin="0.5,0.5" >
      </Ellipse>
</Grid>
</s:TagVisualization>
```

Obrázek 18: Úplný kód pro zobrazení tagu

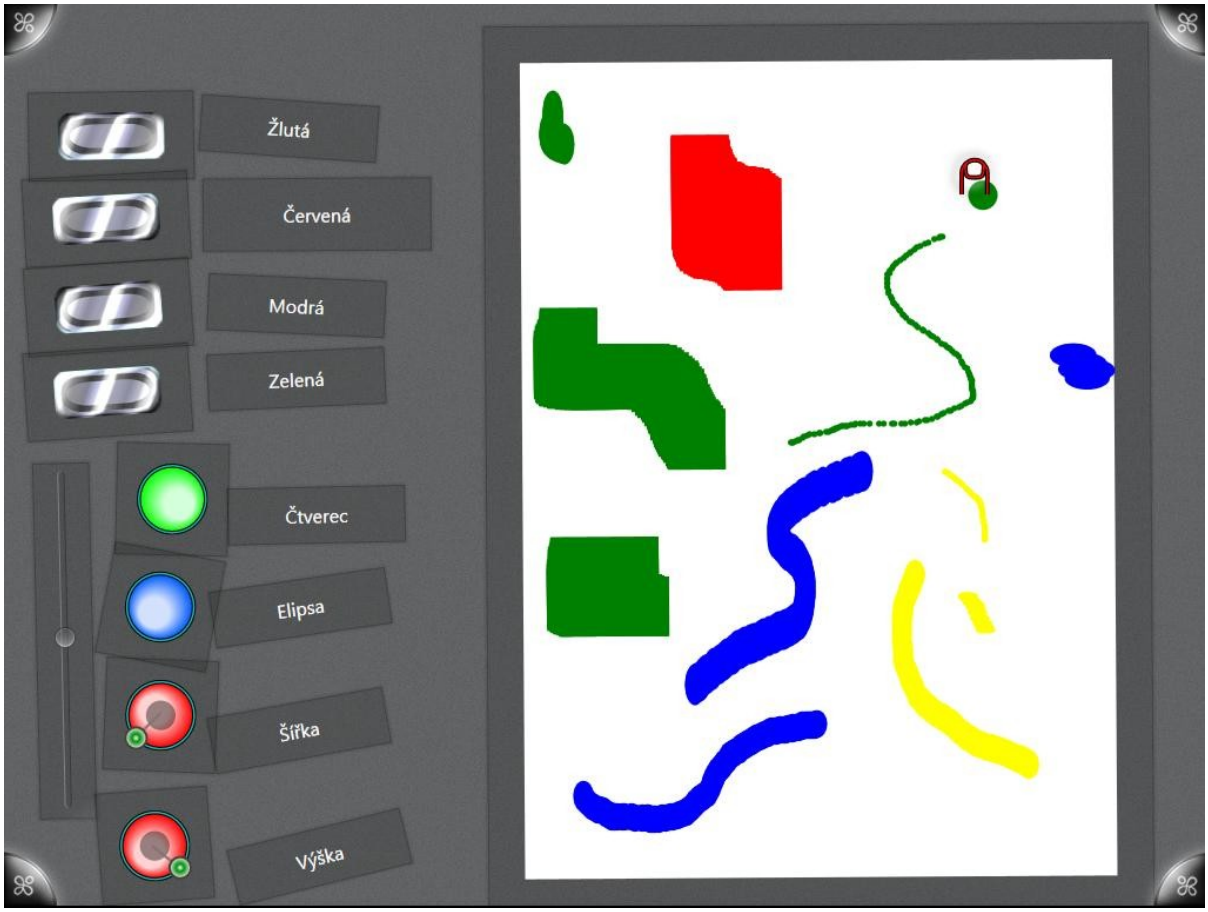
Používání tagů je snadné a brzy ho pochopí i začátečník. Má však jedno velké omezení, je nutné definování tagů provést před inicializací komponenty *TagVisualizer.* Při inicializaci dojde k zapsání definic tagů do slovníku v komponentě a ta se uzavře dalším vstupům. Proto nejde provádět dynamické definování tagů, což může u některých aplikací vadit.

5. Ukázkové úlohy

V rámci seznámení s vývojovým prostředím Microsoft Surface vznikla série aplikací, na kterých si autor textu ověřoval některé postupy vývoje a vlastnosti některých komponent. Tyto aplikace jsou v různé fázi dokončení a mnohé mají stejnou funkci (té však bylo dosaženo různými postupy), proto zde nebudou popsány všechny, ale pouze ty, které splňují většinu pravidel o tom, jak mají aplikace v prostředí Microsoft Surface vypadat. Popis aplikací se omezuje na jejich funkci a na to, jakým způsobem vznikaly určité efekty. V závěru je popsáno, jakým způsobem by bylo možno aplikaci ještě obohatit, případně srovnání s běžnými aplikacemi.

5.1. Aplikace Chaos

Jednou ze základních vlastností Microsoft Surface je jeho neorientovanost, kdy konečné nastavení orientace a polohy ovládacích prvků je na uvážení uživatele. Tato vlastnost je v této aplikaci dotažena do extrému. Všechny komponenty jsou umístěny na ploše náhodně a na uživateli je, aby zjistil, co která komponenta dělá a kam ji umístí. Celá aplikace je vlastně puzzle, kde uživatel neskládá obrázek, ale uživatelské rozhraní. Po složení celé aplikace uživatel zjistí, že se jedná o obyčejnou aplikaci typu malování a že funkce upravují styl kreslené čáry.



Obrázek 19: Aplikace Chaos

Celá aplikace je postavena na vlastnostech kontejneru *ScatterView*. Všechny komponenty jsou do něj umístěny a tím jsou jim zajištěny funkce manipulace pomocí kontaktů. Samotné komponenty by však byly jen těžko manipulovatelné, protože jejich okraj by byl nepatrný, proto je každá umístěna do kontejneru *ScatterViewItem,* který tak definuje velikost okraje. Způsob zavedení okraje je vidět na uvedeném příkladu, stejně tak i to, že některé prvky mají zakázáno měnit velikost.

```
<s:ScatterViewItem Background="Transparent"
      CanScale="False" Width="95" Height="95">
      <s:SurfaceCheckBox Name="check1" />
</s:ScatterViewItem>
```

Obrázek 20: Umístění komponenty do *ScatterViewItem*

Základní komponentou v kontejneru je *Canvas*. Jelikož jde o běžnou komponentu, plátno nemá funkce pro rozpoznání kontaktů. Proto je nutné tyto události připojit. Bez nich by totiž nebylo možno na plátno kreslit. Jak to provést je uvedeno v následujícím příkladě.

```
<s:ScatterViewItem Height="256" Width="342"
      MinWidth="225" MaxWidth="1000" Background="Transparent">
    <Viewbox x:Name="view" StretchDirection="Both" Stretch="Fill" >
      <Canvas x:Name="canvas" Height="256" Width="342"
            Background="White"
            s:Contacts.ContactDown="OnCanvasContactDown"
             s:Contacts.ContactChanged="OnCanvasContactChanged"/>
    </Viewbox>
</s:ScatterViewItem>
```

Obrázek 21: Plátno *Canvas* s připojením událostí typu kontakt

Přidělením událostí však práce nekončí. Aby bylo možno na plátno kreslit, je napřed vytvořit funkce, které budou přidělovat plátnu tvar, kterým se má kreslit, jeho rozměry, barvu a další vlastnosti. Teprve pak bude na plátno možno něco nakreslit. V základním nastavení je na plátno možno kreslit elipsou, která připomíná otisk prstu. Jakým způsobem to provádět, je popsáno v následujícím příkladu.

```
ellipse = contact.GetEllipse(canvas);
switch (choose_colour)
{
     case 0: ellipse.Fill = Brushes.Blue; break;
     case 1: ellipse.Fill = Brushes.Red; break;
     case 2: ellipse.Fill = Brushes.Green; break;
     case 3: ellipse.Fill = Brushes.Yellow; break;
     default: ellipse.Fill = Brushes.Blue; break;
}
if (check1.IsChecked == true)
   ellipse.Height = slider.Value;
if (check2.IsChecked == true)
   ellipse.Width = slider.Value;
canvas.Children.Add(ellipse);
```

Obrázek 22: Funkce pro kreslení elipsou

Microsoft Surface se nemusí v případě malování omezovat jen na upravený *Canvas.* Druhou možností by bylo použít komponentu, přímo určenou ke kreslení v této platformě. Touto komponentou je *SurfaceInkCanvas.* Tato komponenta obsahuje vše potřebné pro rozpoznávání kontaktů a svým provedením se moc neliší od běžného *Canvasu.* Na rozdíl od něj netřeba definovat žádné pomocné funkce pro kreslení, komponentu stačí umístit a už se s ní dá pracovat. Také umožňuje kontakt identifikovat, což má za následek, že jednotlivé kontakty jsou nezávislé (lze například malovat dvěma barvami současně, což v upraveném *Canvasu* nelze). Běžného *Canvasu* bylo použito z důvodu názorné ukázky, že i běžné komponenty se dají v prostředí Microsoft Surface využít, i když někdy ne s takovým efektem.

Při použití *SurfaceInkCanvasu* v této aplikaci se ukázalo, že tato komponenta má problematicky řešený způsob změny tvaru kreslící čáry. V základním nastavení každý kontakt na plátno způsobí čáru, jejímž základem je elipsa. Komponenta se tímto snaží přiblížit realitě, kdy kreslení prsty po papíře způsobuje stejně tvarovanou čáru. Problém nastává v případě, když chce vývojář styl čáry změnit nebo upravit jeho velikost, jako se to dělá v případě této aplikace. *SurfaceInkCanvas* má přímou podporu změny barvy čáry, změna tvaru je však mnohem komplikovanější. I z tohoto důvodu bylo nakonec rozhodnuto o použití běžné komponenty *Canvas.*

Mimo plátna *Canvas* bylo v aplikaci použito komponent *SurfaceButton, SurfaceCheckBox*, *SurfaceRadioButton* a *SurfaceSlider.* Všechny tyto komponenty jsou odvozeny ze známých komponent a vesměs mají stejnou funkčnost. Výjimkou je pouze *SurfaceSlider,* který umožňuje nejen potáhnutí jezdce, ale i jeho pošťouchnutí (naznačeným pohybem se jezdec rozjede a zastaví o kus dál). Tyto komponenty mění parametry kontaktu, jímž se kreslí na plátno. *SurfaceButton* mění jeho barvu, *SurfaceRadioButton* jeho tvar a *SurfaceCheckBox* v kombinaci se *SurfaceSlider* mění rozměry kontaktu. Díky těmto funkcím lze vybrat, zda kreslit běžným kontaktem, který svým tvarem připomíná otisk prstu, nebo si vytvořit vlastní, například kontakt připomínající psaní perem. Vzhled většiny použitých komponent byl upraven pomocí *templates,* jež byly definovány pomocí nástroje Microsoft Expression Blend. Vzhled komponenty byl upraven tak, aby budil zdání reality a měl odezvu na každý kontakt.

Tato aplikace byla zaměřena hlavně na možnost upravit uživatelské rozhraní podle svého. Tím funkce malování trochu zapadla do pozadí, což se projevilo tím, že neplní některé základní vlastnosti, jež by taková aplikace měla mít. To by mělo být náplní budoucího vylepšení. Na škodu by možná nebylo začít úplně od znovu a vytvořit přímo aplikaci zaměřenou na malování.

Malování v prostředí Microsoft Surface je obecně mnohem přirozenější než v běžných aplikacích díky tomu, že umožňuje kreslit takzvaně „od ruky.“ Otázkou zůstává, zda kreslení pouze pomocí ruky bude současnému uživateli stačit. Běžné aplikace totiž nabízejí i různé funkce kreslení pomocí primitiv, které malování ještě víc ozvláštňují, a zatím žádná aplikace Microsoft Surface takové funkce nepředvedla.

5.2. Ukázková aplikace Tags

Další z ukázkových aplikací, která má přiblížit funkčnost Microsoft Surface, je aplikace Tags. Celá je postavena na manipulaci s tagy, což dělá její ovládání velmi prostým. Pod tagem s hodnotou 192 (Cx00) se skrývá tak zvaný selektor, který umožní vybrat jeden z přednastavených motivů. Tento motiv se po kontaktu zobrazí. Jednotlivé motivy lze zavolat i samostatně, bez použití selektoru, položením tagu s hodnotou 193 (Cx01) až 198 (Cx06).

Obrázek 23: Výběr motivu v aplikaci Tags

Jedná se o první aplikaci pro Microsoft Surface, kterou autor tohoto textu vytvořil, proto je napsána hodně vyčerpávajícím způsobem. Jednotlivé motivy mají samostatné soubory *xaml*, které se pak zvlášť volají pomocí URI. Pokud je použitých tagů málo, není to takový problém. Vývojář by však měl vědět, že Microsoft Surface umožňuje určitou míru generalizace. Té je naplno využito až v aplikaci SurfaceShop.

Postup, jakým se tagy vytvářejí, byl již vysvětlen dříve, proto jen stručně. Aby bylo možno tagy používat, je třeba pracovat s komponentou *TagVisualizer*. Ta bude tagy v aplikaci číst a zobrazovat. Před inicializací komponenty se musí nadefinovat všechny tagy, které budou v aplikaci rozpoznatelné. Definice musí minimálně obsahovat hodnotu, kterou má *TagVisualizer* kontrolovat, a URI adresu na komponentu *TagVisualization,* která se má zobrazit, pokud dojde k nalezení dané hodnoty.

Příklad definuje tag s hodnotou 193 (00xC1), pro kterou se zavolá komponenta v souboru *Tag1.xaml.* Mimo tyto nejdůležitější atributy je v příkladu zaznamenáno, že tag se má zobrazit přímo v místě kontaktu a má být orientován stejně, jak byl položen. Po odebrání tagu má počkat jednu vteřinu a pak pozvolna zmizet. Poslední informace o tomto tagu je, že v aplikaci se může vyskytovat maximálně jeden tag se stejnou hodnotou.

```
ByteTagVisualizationDefinition tagVisualDefinition1 =
      new ByteTagVisualizationDefinition();
tagVisualDefinition1.Value = (byte)193; // C1
tagVisualDefinition1.Source = new Uri("Tag1.xaml", UriKind.Relative);
tagVisualDefinition1.PhysicalCenterOffsetFromTag = new Vector(0, 0);
tagVisualDefinition1.OrientationOffsetFromTag = 0.0;
tagVisualDefinition1.LostTagTimeout = 1000.0;
tagVisualDefinition1.MaxCount = 1;
tagVisualDefinition1.TagRemovedBehavior = TagRemovedBehavior.Fade;
Visualizer.Definitions.Add(tagVisualDefinition1);
```

Obrázek 24: Definování motivu na základě tagu

Co se týká komponenty, která se má po rozpoznání tagu zobrazit, platí stejná pravidla jako pro všechny komponenty v prostředí Microsoft Surface. Jediným rozdílem je skutečnost, že celá komponenta musí být umístěna v kontejneru *TagVisualization.* Bez něj by byla komponenta brána jako obyčejný prvek, který se nedá v prostředí *TagVisualizer* zobrazit. Tvorba jednotlivých motivů je velmi prostá. Prakticky se jedná jen o dva prvky typu *Ellipse,* které jsou dále upravovány. Selektor naproti tomu tvoří šest tlačítek *SurfaceButton,* jejichž jednotlivé styly byly upraveny tak, aby připomínaly motiv, který budou volat. Motiv je pak umístěn do kontejneru *ScatterView,* kde s ním lze manipulovat jako s každým obyčejným prvkem.

Při tvorbě této aplikace nastal jistý problém, který bylo nutno řešit. Kontejnery *ScatterView* a *TagVisualizer* tvořily dvě nezávislé třídy, které mezi sebou neměly žádnou vazbu. *ScatterView* tak nevěděl, který prvek v selektoru byl vlastně vybrán a který má zobrazit. Bylo třeba vytvořit delegáta, který by byl přístupný oběma třídám a obsahoval všechny důležité prvky (v tomto případě šlo jen o číslo prvku, který se měl zobrazit). Ten je volán vždy, když je stisknuto tlačítko selektoru, tedy vždy když došlo k volání motivu. V kontejneru *TagVisualizer* se pak volá událost, která nastane vždy, když se má zobrazit daná komponenta. Událost nese název *VisualizationAdded* a je vlastní komponentě *TagVisualizer.* Způsob umístění motivu do kontejneru *ScatterView* je vidět v následujícím příkladu.

```
ScatterViewItem container = new ScatterViewItem();
container.CanScale = false;
container.Center = e.ValidationCenter;
container.Orientation = e.ValidationOrientation;
switch(e.SelectNumber)
{
    case 1:
        {
            Tag1 newPod = new Tag1();
            container.Content = newPod;
            break;
        }
}
```

Obrázek 25: Výběr motivu na základě vybrané hodnoty

Z toho, že se motiv zobrazí je vidět, že obsahem kontejneru *ScatterView* může být opravdu cokoliv. I když je daný motiv třídy *TagVisualization,* kontejner ho příjme a pracuje s ním jako s běžným objektem *ScatterViewItem.*

Aplikace tohoto typu se v běžném provozu nevyskytují, protože schopnost rozpoznávat objekty na základě tagů je vlastní pouze platformě Microsoft Surface. Základní idea, se kterou byla tato aplikace vyvíjena, je vytvořit aplikaci, která by se dala použít v hospodách či kavárnách. Při umístění sklenice či jiného předmětu na desku stolu by si zákazníci mohli vybrat „virtuální podtácek." Jiné využití aplikace postavené výhradně na přítomnosti tagu nemají. Ačkoliv z tohoto popisu se může zdát, že používání tagů je složitá operace, kterou nový uživatelé nemusí hned na poprvé zvládnout, opak je pravdou. Tato problematika je dobře zvládnuta a každý uživatel o ní jistě najde spoustu informací.

5.3. SurfaceShop

Cílem této aplikace bylo ověřit, může-li stolek Microsoft Surface obstát v reálném obchodě. Z množství možností, které může stolek obchodu nabídnout, byla vybrána aplikace, která se svým vzhledem podobá běžným webovým e-shopům. Jejím účelem je představit zákazníkům veškerý sortiment zboží, kterým obchod disponuje.

Obrázek 26: SurfaceShop

5.3.1. Základní analýza

Základním účelem webových aplikací známých jako e-shopy je nabídnout potenciálnímu zákazníkovi sortiment zboží s možností prohlížet jednotlivé položky a v případě zájmu je objednávat. Důraz je přitom kladen na funkce zaručující bezpečné objednávání produktů. Princip aplikace pro Microsoft Surface je jiný. Předpokládá se, že zákazník bude SurfaceShopu využívat již v obchodě. Funkce objednávání je tak zatlačena do pozadí. Naopak je zde zvýrazněna funkce zobrazení obsahu. Zřetel se také klade na zvýraznění schopností Microsoft Surface.

Před zahájením samotného vývoje aplikace se provedla krátká analýza běžného webového e-shopu a jeho funkce se srovnaly s možnostmi Microsoft Surface. Na základě této analýzy se pak přistoupilo k samotnému návrhu aplikace. E-shopy všeobecně obsahují velké množství produktů. Skutečnost, že se nemusí omezovat kapacitou skutečného obchodu, dává firmám možnost nabízet i zboží, které zatím fyzicky nevlastní. Avšak velké množství produktů volá po určité míře setřídění. Běžné e-shopy proto dělí své produkty do kategorií. Kategorie lze do sebe zanořovat a vytvářet tak strom, jehož listy jsou jednotlivé produkty.

Protože e-shop pro Microsoft Surface bude obsahovat reálná data, i zde bude platit předpoklad velkého množství produktů. Aplikace tedy bude muset být schopna produkty roztřídit do kategorií a jednotlivé kategorie efektivním způsobem zobrazovat. Na škodu nebude ani další funkce běžného e-shopu, vyhledávání. Microsoft Surface má nad webovými aplikacemi výhodu velké pracovní plochy, na které může informace zobrazovat. Může tak uživateli navíc nabídnout i možnost porovnání dvou produktů, což běžné webové e-shopy neumějí.

Předpoklad, že se aplikace bude nacházet přímo v obchodě, sebou přináší ještě jednu funkci, kterou běžné weby nemohou nabídnout. A to je možnost vyhledávání na základě produktu. Použitím tagů lze všechno zboží označit a v případě, že produkt bude položen na desku stolu, zobrazit některé jeho informace. Příkladem z praxe může být známé skenovací zařízení v supermarketech, které na požádání zjistí cenu výrobku.

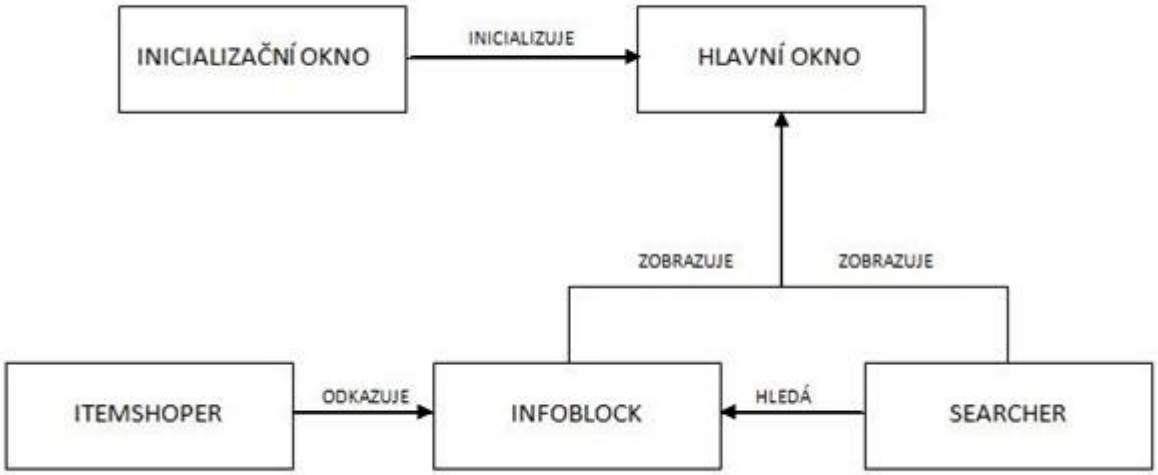
5.3.2. Návrh aplikace SurfaceShop

Prvním problémem, který je třeba vyřešit, je způsob, jakým si bude aplikace SurfaceShop obstarávat reálná data k práci. Nejoptimálnější možností se jeví obstarávání dat z Internetu, ke kterému má stolek připojení. Tím bude zajištěna i nezávislost na typu obchodu.

Webové aplikace typu e-shop používají formát XML, který jim umožňuje přenášet data mezi sebou. Formát se jmenuje XML feed a je ideální pro nasazení reálných dat do aplikace SurfaceShop. Protože data musí být do jednotlivých komponent aplikace zavedeny ještě před inicializací, je třeba vytvořit inicializační okno. V něm si bude moci uživatel určit, jaký druh zboží se má v aplikaci zobrazovat. XML feed se hodí pro přenos mezi weby, ale pro samotné zobrazování produktů je nevhodný. V prvotní verzi aplikace bylo XML feedu použito i k zobrazování, tento způsob však byl nakonec zavrhnut, protože neumožňoval efektivní způsob vyhledávání v kategoriích a zbytečně aplikaci zpomaloval.

V nynější verzi je XML feedu použito tak, že jeho obsah se přepíše do databáze a rozdělí se mezi jednotlivé vytvořené kategorie. Použití databáze umožňuje efektivnější způsob vyvolávání jednotlivých kategorií a výpis produktů je tak mnohem rychlejší. Přepis do databáze trvá poměrně dlouho, proto by neměl nastávat při každém spuštění aplikace. Proto byla aplikace vybavena

inicializačním oknem, které se objeví před vstupem do samotného SurfaceShopu. V tomto okně lze vybrat, odkud se mají data brát. Pokud je vybrán XML feed, dojde k přepisu do databáze. Pokud si uživatel vybere jako zdroj dat databázi, použijí se data, kterými je už naplněna.



Obrázek 27: Blokové schéma aplikace *SurfaceShop*

Protože kategorie mohou i po rozdělení obsahovat velký počet položek, je navrženo menu, v němž bude možno vybírat jednotlivě položky samostatně. Vybrané položky se zobrazí na ploše ve formě *InfoBlocku.* Ten bude obsahovat ty nejdůležitější informace o produktu a budou na něj aplikovatelné nejdůležitější vlastnosti Microsoft Surface (možnost měnit velikost, umístění, rotaci). Pro vyhledávání položek bude v okně k dispozici vyhledávač. Měl by být posunutelný do libovolné pozice a měl by být schopen vyhledávat jak jednotlivé položky, tak celé kategorie.

Rozvržení jednotlivých komponent po ploše by mělo být na uvážení uživatele. Výjimkou je menu, které by mohlo svými rozměry působit potíže, proto je fixně umístěno. Aby však bylo dostupné z kterékoliv strany stolu, bude mít aplikace možnost změnit svou orientaci a natočit se směrem, který uživatel bude považovat za vhodnější. Jiné komponenty by měly být přesunutelné podle uvážení uživatele.

Microsoft Surface bude mít ještě jeden způsob, jakým zobrazovat informace o produktu. Tento způsob je realizovatelný pouze v obchodě, který pracuje již s reálným zbožím. Jedná se o využití tagů, kdy každý objekt v obchodě bude označen tagem, takže po položení produktu na desku stolu se zobrazí *InfoBlock*, který daný produkt popisuje.

5.3.3. XML feed

Tento XML soubor obsahuje všechny podstatné informace o produktech firmy. Využívá se hlavně pro tvorbu e-shopu, protože přehlednou formou vypisuje všechny informace, které zákazník bude chtít o produktu znát. Formát XML feed je standardizovaný, což umožňuje firmám, které nemají prostředky na to, aby si vytvořily vlastní e–shop, využít služeb větších celků, jako je například Zboží.cz či Aukro.com.

Povinnými elementy v tomhle XML souboru jsou SHOP, SHOPITEM, PRODUCT, URL, PRICE_VAT nebo PRICE a VAT. Další elementy uvedené v následujícím seznamu jsou volitelné a lze je z XML souboru úplně vypustit [9].

SHOP - kořenový element, v souboru se vyskytuje jen jednou
SHOPITEM - element obsahuje informace o konkrétním produktu, v souboru je obsažen tolikrát, kolik je produktů
PRODUCT - obsahuje název produktu, je využíván ve fulltextovém vyhledávání
DESCRIPTION - popis produktu
URL - URL adresa produktu
IMGURL - URL adresa na obrázek produktu
PRICE - cena produktu bez DPH, pokud nejsou uvedeny autorské a recyklační poplatky v položce DUES, musí být přičteny k hodnotě této položky
PRICE_VAT - cena produktu s DPH, pokud nejsou uvedeny autorské a recyklační poplatky v položce DUES, musí být přičteny k hodnotě této položky
DUES - autorské a recyklační poplatky, pokud již nejsou započítané do ceny v elementu PRICE nebo PRICE_VAT
VAT - sazba DPH (možné formáty jsou: 19, 0.19 nebo 0,19)
MANUFACTURER - název výrobce produktu
CATEGORYTEXT - zařazení produktu do kategorie, uvádí se všechny kategorie od nejobecnější po nejkonkrétnější, viz. příklad:

SPRÁVNĚ: Elektronika | Mobilní telefony | Příslušenství | Nabíječky
ŠPATNĚ: Nabíječky

Zbytek seznamu se v této aplikaci nepoužívá.

EAN - EAN kód produktu
PRODUCTNO - výrobní číslo produktu udávané výrobcem, u knih např. ISBN, nejedná se o katalogové číslo nebo interní značení produktu
WARRANTY - délka záruky v měsících, pouze celočíselná hodnota
DELIVERY_DATE - doba doručení produktu ve dnech, dodací doba musí být uváděna jako doba od přijetí platby do expedice zboží
SELEKT - tento element slouží k vybrání produktu. E-shop takový produkt zvýrazní, čímž lze upozornit například na zlevněné zboží. Možné hodnoty jsou Y, pokud se jedná o zvýrazněný produkt nebo N, pokud se o zvýrazněný produkt nejedná.

Informací z XML se používá pro zápis do menu a také pro zobrazení jednotlivých produktů. Prvotní verze aplikace nepředpokládala existenci databáze, a proto se veškeré informace předávali pomocí XML formátu. Tento formát byl zachován i po připojení databáze, protože položka ke svému zobrazení potřebuje informace z více zdrojů a XML formát je v tomto ohledu rychlejší než postupné dotazování databáze.

Aplikace pro Microsoft Surface zpracovává XML feed stejným způsobem jako webové e-shopy. Na základě informací z XML souboru se provede zápis do databáze, z které se pak informace zobrazují. Během implementace se ukázalo, že samotný formát XML feed nestačí pokrýt všechny požadavky aplikace. XML a potažmo databáze neobsahuje žádné dodatečné informace typu tagy a adresy dalších možných obrázků. Proto byl k projektu připojen ještě jeden dokument XML, který tyto informace obsahuje.

Tento soubor obsahuje to nejnutnější pro tvorbu tagů a informačních panelů s dodatečnými obrázky. Zdrojem obrázků může být lokální zdroj, jako je vidět v příkladu, nebo seznam URL adres, kde se obrázky nacházejí.

```
<ROOT>
  <ITEM>
    <URL>http://www.fissler-hrnce.cz/fissler-finecut/d-72899/</URL>
    <TAG>192</TAG>
    <VIEW>
      <PICTURE>http://www.fissler-hrnce.cz/fotocache/bigorig/1030004.jpg</PICTURE>
      <PICTURE>http://www.fissler-hrnce.cz/fotocache/bigorig/1070000820.jpg</PICTURE>
      <PICTURE>http://www.fissler-hrnce.cz/fotocache/bigorig/1169310.jpg</PICTURE>
      <PICTURE>http://www.fissler-hrnce.cz/fotocache/bigorig/1612004.jpg</PICTURE>
      <PICTURE>http://www.fissler-hrnce.cz/fotocache/bigorig/831603.jpg</PICTURE>
    </VIEW>
  </ITEM>
</ROOT>
```

Obrázek 28: XML soubor pro dodatečné informace o produktech

5.3.4. Databáze SQLite

Samotný formát XML je pro zobrazení většího počtu prvků nevhodný. Neumožňuje rozdělení do kategorií a vyhledávání v něm trvá poměrně dlouho. Proto se běžně e-shopy ukládají do databází a XML feed slouží pouze k přenosu mezi nimi.

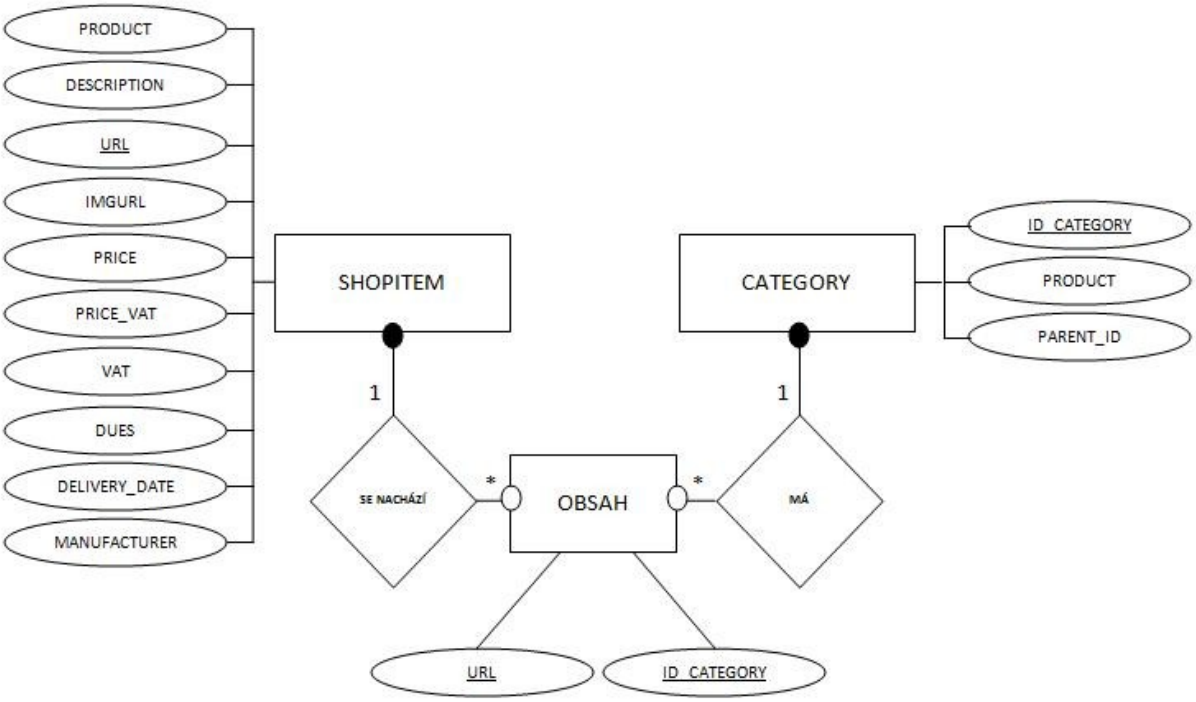
Tato aplikace využívá databáze SQLite. Databáze je volně dostupná a nevyžaduje složitější provozované systémy řízení báze dat (SŘBD). Jejím největším přínosem je rychlost, která je vyšší než u většiny databázi typu klient-server, malý prostor, který databáze zabírá, efektivnější práce s pamětí systému a v neposlední řadě i schopnost databázi šifrovat. To všechno je dostupné v jediné knihovně, v níž netřeba nic instalovat, nastavovat nebo připojovat, Přitom nabízí funkcionalitu klasického SŘBD postaveného na platformě SQL, tedy dotazování, filtrování, aktualizace dat a jiné [10].

Databáze pro aplikaci SurfaceShop obsahuje tři základní tabulky: *SHOPITEM, CATEGORY* a *OBSAH*. Do nich je přepsán celý obsah souboru XML feed. První tabulka slouží k ukládání všech produktů, které se v prostředí e-shopu vyskytují. Jednotlivým sloupcům tabulky odpovídají elementy XML feed souboru. Primární klíčem je element *URL,* který je pro všechny položky povinný a jedinečný.

Tabulka *CATEGORY* slouží k ukládání informací o všech nalezených kategoriích. Má tři sloupce, z nichž *ID_CATEGORY* je primární klíč, *PRODUCT* zaznamenává název kategorie a *PARENT_ID* kontroluje, zda kategorie je na nejvyšší úrovni nebo má nějakého předka. Tím lze vytvářet i zanořené seznamy kategorií. Poslední tabulka propojuje *SHOPITEM* a *CATEGORY*. Obsahuje jen dva sloupce *ID_CATEGORY* a *URL*.

Pro každou položku souboru XML feed se provede zápis do databáze. Napřed se zapíšou všechny údaje do tabulky *SHOPITEM* a pak se provede rozdělení do kategorií. Před zápisem vyhledané kategorie do tabulky se zkontroluje, zda už daná kategorie není vytvořena. To se dá ověřit vyhledávacím dotazem. Pokud nebyla kategorie nalezena, neexistuje a lze ji do tabulky zapsat. Zároveň se zápisem kategorie do tabulky je třeba zjistit, jaké identifikační číslo nová kategorie

dostala. Tohle číslo totiž bude potřeba pro zápis podřízené kategorie nebo pro zápis položek do tabulky *OBSAH*.



Obrázek 29: Diagram databáze

V té se zapisují všechny položky s posledními vzniklými kategoriemi. Tato tabulka je potřebná, aby se dalo ze seznamu kategorií volně přejít na seznam položek v kategorii. Během vývoje aplikace se v tomhle bodě vyskytnul problém. Aplikace předpokládá, že jednotlivé kategorie budou definovány následně:

```
<CATEGORYTEXT>Kuchyňdoplňky | Baterieprokuchyně</CATEGORYTEXT>
<CATEGORYTEXT>Vodovodbaterie | Kuchyňbaterie</CATEGORYTEXT>
```

Obrázek 30: Korektně fungující definice kategorií

Produkt může být i ve více kategoriích, na což aplikace pamatuje. Daná položka se tak zapíše do kategorie *Baterieprokuchyň,* která je podkategorií kategorie *Kuchyňdoplňky* a také do kategorie *KuchyňBaterie,* která je podkategorií *VodovodBaterie.* Tabulka *OBSAH* má tak dva nové záznamy. Protože však ve formátu XML feed není zcela jasno, jak psát kategorie, lze se setkat i s následující konstrukcí, která však povede k chybě:

```
<CATEGORYTEXT>Vodovod</CATEGORYTEXT>
<CATEGORYTEXT>Vodovod | Pákovbaterie</CATEGORYTEXT>
<CATEGORYTEXT>Vodovod | Pákovbaterie | Vanovébaterie</CATEGORYTEXT>
```

Obrázek 31: Nekorektně fungující definice kategorií

Aplikace správně vytvoří kategorie, ale současně do každé z nich vloží i produkt, ve kterém se taková konstrukce vyskytuje. Celkově se tedy v tabulce *OBSAH* objeví produkt třikrát, ačkoliv správně by záznam měl být jen jeden v kategorii *Vanovébaterie.* Redundance způsobené touto nesrovnalostí ve standartu způsobuje zbytečné prodlevy při vyhledávání položek a v ojedinělých případech i ukončení přepisu do databáze.

Po spuštění aplikace bude z databáze načtena první úroveň kategorií, tedy ty, které neobsahují předka. Výsledek dotazu se pošle do menu jako zdroj dat.

```
DataSet myDataSet = new DataSet();
sqlite_cmd.CommandText =
      "SELECT * FROM CATEGORY WHERE PARENT_ID = 0 ORDER BY PRODUCT";
SQLiteDataAdapter adapter = new SQLiteDataAdapter();
adapter.SelectCommand = sqlite_cmd;
adapter.Fill(myDataSet, "Items");
```

Obrázek 32: SQL dotaz pro první úroveň kategorií

Na stejném principu se pokračuje i v dalších úrovních zanoření s tím rozdílem, že *PARENT_ID* už nebude 0, ale hodnota vybrané kategorie. Pokud v kategorii nedojde k nalezení dalších kategorií podřízených vybranému číslu, přejde se do tabulky *OBSAH* a provede se vyhledávání produktů nad danou kategorií. Stejným způsobem se provádí dotazování na položky.

5.3.5. Looping menu a InfoBlock

Aby uživatel mohl jednoduchým způsobem postupovat systémem kategorií a položek, bylo vytvořeno *Looping menu.* To slouží k zanořování se do úrovní kategorií a pro výběr položek, které se mají zobrazit. Jednou z vlastností Microsoft Surface je dynamičnost, proto bylo menu upraveno tak, aby odpovídalo tomuto požadavku. Běžné menu lze prolistovat od začátku do konce a pro návrat na začátek musí uživatel použít posuvné lišty. *Looping menu* nic takového nemá, protože po poslední položce menu následuje zase první a celý seznam se opakuje. Seznam zboží se tak opakuje ve věčné smyčce, odtud název *Looping menu.*

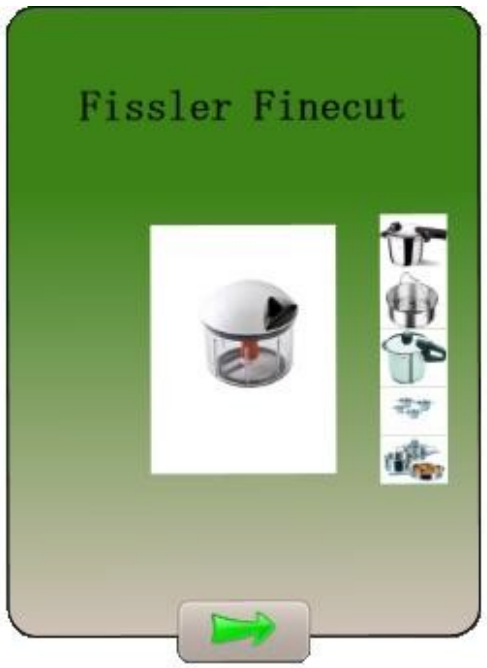
Základem pro toto menu je komponenta *SurfaceListBox.* Ta má základní vlastnosti odpovídající komponentě Microsoft Surface (například schopnost posouvat obsah šťouchnutím). Na *SurfaceListBox* byl aplikován styl, který upravuje jak vzhled, tak chování komponenty.

```
<Setter Property="Template">
  <Setter.Value>
    <ControlTemplate>
      <s:SurfaceScrollViewer
         VerticalScrollBarVisibility="Hidden"
         CanContentScroll="true">
         <sc:LoopingMenu IsItemsHost="True"/>
      </s:SurfaceScrollViewer>
    </ControlTemplate>
  </Setter.Value>
</Setter>
```

Obrázek 33: Definování stylu pro *SurfaceListBox*

Pro zobrazování položek z menu je použito standardního kontejneru *ScatterView*. Položka *ScatterViewItem* je však upravena, takže každý objekt, který se z menu vyvolá, je typu *InfoBlock*. Tento blok obsahuje základní informace o vybraném produktu a z počátku je malý, uživatel ho však může kdykoliv roztáhnout.

InfoBlock obsahuje jen základní informace o produktu. Pro zobrazení je určen jenom název, obrázek produktu, jeho cena, případně poznámka, která stručné popisuje některé doplňující informace, o kterých je obchod přesvědčen, že by je měl zákazník znát. V případě, že tento produkt má odpovídající záznam v doplňujícím XML souboru, jsou zde navíc zobrazeny další možné náhledy na produkt. Tyto informace patří mezi uživateli nejhledanější a nabízet jich víc by znamenalo výrazně zvětšit *InfoBlock*. I tak je zobrazení všech informací problematické a je řešeno tím způsobem, že je poznámka a cena produktu skryta a vyvolá se teprve až po stisknutí tlačítka. Animace se postará, aby uživatel podlehl iluzi, že se *InfoBlock* obrátil na druhou stranu.



Obrázek 34: *InfoBlock*

V původním návrhu se chtěl autor vyhnout používání tlačítka a animace. Dodatečné informace se měly aktivovat systémem „stiskni a drž.“ Uživatel měl přistoupit ke komponentě, stisknout například obrázek a *InfoBlock* by zvětšil svůj rozměr a odhalil tak další údaje. Tento způsob byl ale nakonec zavržen, protože by vyžadoval použití dodatečných událostí pro komponentu *Image* z třídy *s:Contacts*. Po kontaktu s obrázkem se sice *InfoBlock* zvětšil, nebyl se však již schopen opět zmenšit. Problém je v tom, že komponenta *Image* umístěná v *ScatterViewItem* nedokáže vyvolat událost *s:Contacts.ContactUp*. *ScatterViewItem* má totiž tuto událost vyhrazenou pro umístění a uložení polohy objektu v kontejneru *ScatterView*. Proto podřízené objekty nemohou tuto událost volat.

Možným řešením podobných problémů je používat pouze operace *s:Contacts.ContactDown,* která má v tomto případě výjimku, nebo se uchýlit k použití aktivního prvku, například tlačítka, který má své vlastní události. Druhé možnosti bylo nakonec použito i v této aplikaci. Způsob, jakým se spouští animace pomocí tlačítka, je objasněn v následujícím příkladě.

```
Storyboard sbOpening = this.Resources["Back_Flip"] as Storyboard;
if (sbOpening != null)
{
      sbOpening.Begin();
}
```

Obrázek 35: Spuštění animace

5.3.6. Vyhledávací panel

Pokud uživatel ví, co hledat, může použít vyhledávacího panelu, aby se k dané položce dostal přímo. Samotné procházení menu může být zdlouhavé, zvláště pokud obsahuje hodně kategorií a podkategorií. Proto byl do aplikace přidán vyhledávací panel, který umožňuje rychlejší vyhledávání.

Obrázek 36: Panel vyhledávání

Vyhledávací panel je tvořen jako uživatelská komponenta umístěna do komponenty *ScatterViewItem.* Tím je zajištěn volný pohyb panelu, takže uživatel ho může odsunout stranou, pokud s ním nehodná pracovat. Panel by měl být do určité míry statický, proto mu byla vypnuta schopnost rotovat a měnit svou velikost.

Samotná funkce vyhledávání se provádí SQL dotazem ve tvaru *SELECT * FROM* [volba_uzivatele] *WHERE (PRODUCT LIKE* [volba_uzivatele]*) ORDER BY PRODUCT.* Jak je patrné, vyhledávací panel nevyhledá jen jeden produkt, ale všechny se stejnou skupinou znaků v názvu produktu. Uživatel si také může zvolit, zda hledat konkrétní položky nebo celé kategorie. Na této volbě záleží, zda se bude hledat v tabulce *CATEGORY* nebo v tabulce *SHOPITEM.*

Výsledky vyhledávání se zpracují stejným způsobem, jako všechny ostatní dotazy týkající se menu. Data set vytvořený jako odpověď na SQL dotaz, je umístěn do *Looping menu* a tím dojde k jeho zobrazení. Až uživatel prohlédne všechny výsledky, menu lze pomocí návratového tlačítka vrátí do původního stavu.

5.3.7. Rozpoznávání produktu podle tagu

Jednou z nejdůležitějších funkcí, které má uživatel k dispozici, je schopnost stolku jednoznačně identifikovat objekty. Princip rozpoznávání byl už popsán výše, a proto jen stručně. Každý produkt ve firmě je vybaven tagem, který ho v rámci aplikace jednoznačně určuje. Zákazník v obchodě má tak možnost vybrat produkt, který ho zajímá a položením na desku stolku o něm získat potřebné informace (název a cenu).

Celá informace se objeví jen na tak dlouho, co je produkt položen na stolku. Po odebrání informace zase zanikne. Základem celé funkce je opět komponenta *TagVisualizer* a definování všech použitých produktů opatřených tagy. To se jeví jako největší problém. XML feed neobsahuje žádné informace pro zadávání tagů, proto se musí doplnit dodatečně. Řešením je použití vnějšího XML souboru, ve kterém jsou tagy produktů definovány. Před inicializací komponenty *TagVisualizer* se volá vnější soubor a všechny položky z něj jsou definovány jako tagy.

Na uvedeném příkladu je vidět, jakým způsobem se určování položky tagu provádí. Předpokladem přitom je, že všechny tagy jsou definovány. Tagy, které by definovány nebyly, nemají možnost tuto událost spustit. Pokud je událost spuštěna, je napřed určena hodnota tagu, který ji vyvolal. Poté se načte dodatečný XML soubor, ve kterém se vyhledá odpovídající záznam. Pomocí záznamu URL se v tabulce *SHOPITEM* vyhledají všechny informace, které jsou pro zobrazení tagu potřeba. Poté už je možno tag zobrazit.

```
Item_Shoper p = e.TagVisualization as Item_Shoper;
XmlDocument reader = new XmlDocument();
reader.Load("Resources/Data.xml");
XmlNodeList seznam = reader.GetElementsByTagName("ITEM");
sqlite_conn = new SQLiteConnection("Data
     Source=Resources/Database.db;");
sqlite_cmd = sqlite_conn.CreateCommand();
foreach (XmlElement node in seznam)
{
     if (p.VisualizedTag.Byte.Value.ToString() ==
             node.SelectSingleNode("TAG").InnerText)
    {
      sqlite_cmd.CommandText =
      "SELECT * FROM SHOPITEM WHERE URL = @URL ORDER BY PRODUCT";
      sqlite_cmd.Parameters.Add("@URL", DbType.String, 250).Value =
      node.SelectSingleNode("URL").InnerText;
      sqlite_datareader = sqlite_cmd.ExecuteReader();
```

Obrázek 37: Zobecněné rozpoznávání tagů

5.3.8. Budoucí vylepšení

Aplikace v současném stavu lze použít v obchodě, kde si zákazníci budou moci prohlížet sortiment nabízeného zboží, produkty z jednotlivých kategorií porovnávat a v případě, že bude tato aplikace umístěna přímo v místě prodeje, i okamžitě vybrané zboží odebrat. Navázání aplikace na reálná data funguje bezchybně, takže lze skutečně SurfaceShop použít jako náhradu běžného e-shopu.

Přesto je zde prostor pro mnohé vylepšení. Přínosem by jistě bylo možnost ukládat informace o zákaznících do databáze. Tím by bylo možno aplikaci rozšířit z prohlížeče zboží na skutečný e-shop, ve kterém by bylo možno nejen zboží objednávat ale přímo i platit. Dále by to umožňovalo vytvářet například zákaznický košík, který by zaznamenával všechny pořízené produkty. Dalším vylepšením by bylo možno pravidelným zákazníkům poskytovat tagy, které by například prováděly zlevnění určitých položek.

Jako všechny ostatní ukázkové aplikace je i tato určena do rukou jednoho uživatele. Panel pro prohlížení produktů je přístupný jen z jedné strany a i když je možno orientaci celé aplikace obrátit, přístup k aplikaci je stále omezen. Jednoduchým řešením z praxe, které funguje, by bylo přidání dalšího panelu na opačnou stranu aplikace. Prostor pro prohlížení produktů by byl menší, ale aplikaci by bylo možno používat dvěma zákazníky z obou stran současně.

Poslední vylepšení, které by aplikace mohla zužitkovat, je možnost upravovat XML feed. S touto funkcí by bylo možno registrovat nové produkty přímo z prostředí stolku, což by bylo jistě přínosné. V současném stavu je totiž podnik vlastnící aplikaci nucen při každé změně v XML feed provést přepis do databáze znovu, což při velkém počtu možných produktů a častých změnách může způsobovat zbytečné prodlevy. Možná by v budoucnu bylo lepší pro aplikace SurfaceShop v prostředí Microsoft Surface vytvořit vlastní formát, protože XML feed tak jak je v současné době definován, tyto aplikace spíše omezuje.

5.4. Použití aplikací Microsoft Surface

Obecně se dá tvrdit, že většinu běžných aplikací se dá vytvořit i v prostředí Microsoft Surface. Rozdíl je pouze v uživatelském rozhraní, o kterém rozhoduje použitá platforma. Protože Microsoft Surface je omezen tím, že nemůže využívat vnějších zařízení, se způsob ovládání aplikace někdy podstatně liší od jeho běžného protějšku, přesto však lze mnohé funkce zachovat a využívat přitom výhod této platformy.

Největší výhodu těchto aplikací vidí autor tohoto textu ne v možnosti používání tagů, ani v možnosti ovládat aplikaci netypickým způsobem, ale v možnosti pracovat s aplikací ve skupině. Své uplatnění proto nachází Microsoft Surface hlavně v oblastech, kde s ním může pracovat co nejvíc lidí, například knihovny, kavárny, prezentační místnosti různých společností, školy a jiné. Aplikace, které umožňují sdílet zážitek z netradičního zařízení, jsou zábavné a nehraje roli, zda se jedná o aplikaci určenou k řízení projektu nebo hru Člověče, nezlob se.

Zde také leží budoucnost aplikací Microsoft Surface. Konferenční stolek je jako dělaný pro aplikace, které by se mohly shrnout do kategorie „společenské.“ Čím víc uživatelů může s danou aplikací pracovat současně, tím je aplikace zábavnější a uživatelé ji hodnotí lépe. Proto i nadále bude tato platforma pronikat na místa veřejně přístupná, kde k nim bude mít přístup co nejvíc lidí. Využití v domácnostech jako takových již tak převratné nebude. Brání tomu jak vysoká cena zařízení tak i to, že aplikace vytvořené pro domácí použití se dají nahradit tradičními prostředky.

6. Závěr

Cílem této práce bylo seznámit se s vývojovým prostředím Microsoft Surface, přehledným způsobem popsat, co použití této platformy obnáší pro tvorbu uživatelského rozhraní a vytvořit kolekci ukázkových příkladů, na nichž by bylo možno ilustrovat základní vlastnosti této platformy. Konečným bodem je shrnutí všech dosažených výsledků a usnesení se na závěru, zda může technologie Microsoft Surface obstát v konkurenci s běžnými aplikacemi.

Po téměř dvouletém působení počítače Microsoft Surface na trhu je toto zařízení stále ještě v mnohých kruzích neznámé. Informace o tomto zařízení jsou kusé a většinou se omezují na popis ovládání ukázkových aplikací. Běžnému uživateli to může stačit, ale jen málo informací se týká samotného vývoje aplikací. Ten je totiž provázen různými neduhy, z nichž některé byly již popsány v tomto textu. Jiné však zmíněny nebyly, proto závěrem této práce autor vytvořil malé shrnutí všech pozitivních i negativních jevů, jež z prostředí Microsoft Surface vyplývají.

Konferenční stolek s dotykovým displejem místo desky nabízí široké možnosti využití a bezesporu má své výhody. Ať už si uživatel vybere jakoukoliv aplikaci pro Microsoft Surface, vždy může očekávat mimořádný zážitek z nevšedního způsobu ovládání. Díky přímé interakci s daty bude uživatele práce s tímto zařízením bavit a díky tomu, že interakce vychází převážně z kontaktů a přirozených gest rukou, si daleko rychleji osvojí ovládání aplikace.

Tento zážitek je ještě víc umocněn vlivem toho, že stolek může obsluhovat několik uživatelů současně. To aplikacím přináší dosud neokoukané možnosti. Uživatelé mohou spolupracovat nad jednou úlohou nebo stát proti sobě a provádět činnosti nezávisle na sobě. Rčení, že sdílený zážitek je silnější, je v tomto případě jednoznačně platné.

Už z toho, že stolek může obsluhovat několik uživatelů, je patrné, že zařízení rozpozná nebývalé množství kontaktů. To také přináší možnost ovládat aplikace netypickými způsoby, které však uživatel shledá přijatelnějšími než běžné způsoby ovládání myší a klávesnicí. Možností použití multidotyku, jak se tato vlastnost často nazývá, se Microsoft Surface liší od běžných dotykových displejů, které jsou schopny rozpoznat jen minimální množství kontaktů. Mimo to také stolek umožňuje rozpoznávat i některé objekty na základě identifikátoru. Tento identifikátor se nazývá tag a také díky němu získává platforma Microsoft Surface výhodu nad běžnými systémy, jež objekty nerozpoznávají. Aplikace se tak dají ještě rozšířit o možnosti manipulace s různými objekty, které se položí na povrch stolku.

Pokud se tyto vlastnosti spojí s líbivým vzhledem aplikací, je jasné, proč uživatele Microsoft Surface považují práci s tímto zařízením za jedinečný zážitek. I vývoj aplikací pro Microsoft Surface je v mnoha ohledech „škola hrou.“ Vývojář jistě ocení, že pro práci s touto platformou se nemusí učit nový programovací jazyk. Celý vývoj je postaven na základech WPF a pokročilý webový vývojář nebude mít potíže do jeho tajů proniknout. Množství nástrojů, které jsou vývojáři k dispozici, mu v tom ještě víc napomáhá. Z pohledu vývoje však Microsoft Surface přece jen ztrácí svůj lesk. Tvorba aplikací je svázána někdy až s nesmyslnými pravidly, které realizaci aplikace stěžují. Některé z problémů již byly zmíněny v předcházejícím textu, ale naneštěstí je jich víc.

Jako největší problém se jeví omezení pro počítače na vývoj aplikací. Platforma Microsoft Surface používá 32 bitovou verzi operačního systému Windows Vista, a proto je na stejnou verzi omezen i operační systém počítače pro vývoj aplikací. Pro mnohé uživatele to však znamená, že musí slevit a přestat používat výkonnější 64 bitovou verzi. Po dvouletém působení na trhu by se přitom dalo čekat, že verze pro 64 bitové operační systémy bude již dávno k dispozici, případně bude vytvořen emulátor, aby vývoj aplikací nebyl tak vázán na operační systém. Až do dnešních dnů se tak nestalo.

Samotné omezení na 32 bitovou verzi by nevadilo, kdyby se mezi podporovanými systémy nacházel nejrozšířenější systém vůbec: Windows Vista Home. Tento typ operačního systému však také není podporován, což nutí další uživatele k zbytečným instalacím.

Autor tohoto textu si myslí, že hlavní vinu za problém požadavků na systém nese Microsoft Surface Simulator. Jeho základem je s nejvyšší pravděpodobností operační systém stolku, obohacený o další funkce. To však znamená, že Microsoft Surface Simulator má stejné požadavky na systém jako samotný stolek. V budoucí verzi by bylo lepší simulátor upravit, aby fungoval nezávisle na použité platformě. Simulátor obecně způsobuje řadu problémů související s nastavením zobrazení. Často si vynucuje změnu natavení rozlišení na hodnotu 1024x768 (nastavení použité ve stolku). Zřídka tak dochází ke vzniku problémů v jiných aplikacích, výjimečně i k pádu systému.

Druhým největším problémem, se kterým se musí začínající vývojář potýkat, je nedostatek zdrojů ke studiu této problematiky. Platforma Microsoft Surface je pořád ještě poměrně nová záležitost a většina informací o ní se spíše omezuje na popis výhod, které přináší, a způsobů, jakým se ovládají jednotlivé aplikace. Uživatel, který chce realizovat vlastní nápady pro toto zařízení, má prakticky k dispozici jen dokumentaci a ukázkové příklady, které získá instalováním Microsoft Surface SDK. Za tímto problémem stojí hlavně cena celého zařízení. Konferenční stolek sám o sobě stojí přes 12 000 $, což není zanedbatelná částka. Samotné SDK stojí něco přes 2 500 $, což také není málo. Běžný vývojář si takový nástroj nejspíš nekoupí, protože nemá možnost si své aplikace v reálných podmínkách vyzkoušet (i autor tohoto textu se s tímto zařízením přímo nesetkal). Proto je vývoj aplikací v rukách firem, které mají dost finančních prostředků na pořízení tohoto stolku. Ty však své aplikace představují jen po uživatelské stránce, takže začínající vývojář nemá možnost zjistit, jaký způsobem dané efekty vznikly.

Ačkoliv má vývoj aplikací pro prostředí Microsoft Surface své nevýhody, lze pomocí něj vytvářet aplikace, které v běžném uživateli budou budit dojem, že pracuje s reálnými daty. Z rozdílných vlastností platformy však vyplývají některé povinnosti, které je nutno dodržovat, aby vzhled a funkčnost aplikace netrpěli. Výsledkem bylo vytvoření série pravidel a rad, hlavně však desatera, které obsahuje nezbytné minimum, které by každý vývojář měl ve svých aplikacích dodržovat. Ukázkové aplikace a hlavně pak SurfaceShop ukázaly, že použití Microsoft Surface se nemusí omezovat jen na určité druhy aplikací. Použití platformy je vskutku všestranné a je jen na schopnostech vývojáře, jakým způsobem toho využije.

Co říci závěrem? Microsoft Surface je bezesporu počinek, který na první pohled zaujme. Nepřítomnost jiného zařízení nevadí a tato platforma se dá použít pro všechny známé typy aplikací, ať už se jedná o aplikace firemní či soukromé. Uživatelé obecně tvrdí, že s aplikacemi pro Microsoft Surface se pracuje mnohem lépe než s jejich běžnými protějšky. Autor tohoto textu má na tohle tvrzení názor, že je to způsobeno hlavně tím, že tyto aplikace jsou vždy uzpůsobeny pro práci více uživatelů současně. Zážitek je vždy silnější, pokud se o něj má uživatel s kým podělit, a je vedlejší, jestli se jedná o práci nad projektem nebo o posezení nad odpočinkovou aplikací. V dnešní době je tento trend patrný v mnoha odvětvích. Zakládají se komunikační weby, které navštěvují miliony návštěvníků (Facebook – 400 milionů aktivních uživatelů) a největší tržby v oblasti počítačových her mají ty firmy, které produkují hry, které se dají hrát ve skupině (světoznámé MMORPG World of Warcraft – dne 20. prosince 2008 11,5 milionu platících zákazníků).

Cena je však přece jen nad možnosti běžného uživatele a tak se toto zařízení bude v budoucnu nejčastěji vyskytovat jen v prezentačních místnostech různých institucí, které si toto zařízení bude moci dovolit. Možná, že v budoucnu, až bude tato technologie v širším povědomí a její cena půjde dolů, si najde cestu i do běžných domácností. Zůstává však otázka, zda konkrétně tato technologie neupadne v zapomnění. Firma Microsoft totiž hodlá již brzy představit následníka Microsoft Surface, který bude mít mimo jiné možnost ovládat aplikaci pomocí gest, při kterých se však nebude nutné povrchu stolu dotýkat. Tato technologie se zatím skrývá pod prozatímním názvem Second Light [11].

7. Reference

[1] *Microsoft Surface – Wikipedia, free encyklopedia* [online].
URL:<http://en.wikipedia.org/wiki/Microsoft_Surface>

[2] *What lurks below Microsoft's Surface? A brief Q&A with Microsoft* [online].
URL:<http://arstechnica.com/gadgets/news/2007/05/what-lurks-below-microsofts-surface-a-qa-with-microsoft.ars>

[3] *Microsoft Surface : Behind-the-Scenes First Look (with Video) – Popularmechanics* [online].
URL:<http://www.popularmechanics.com/technology/gadgets/news/4217348?page=2>

[4] *Welcome to Microsoft Surface* [online].
URL:<http://www.microsoft.com/surface>

[5] *Microsoft Surface: udělá na vás dojem – Zdroják* [online].
URL:<http://zdrojak.root.cz/clanky/microsoft-surface-udela-na-vas-dojem>

[6] MICROSOFT SURFACE SDK 1.0, WORKSTATION EDIT. *Microsoft Surface Interaction Design Guidelines* [instalační balíček SurfaceSDKWE.msi]. Microsoft Corporation, 2008.

[7] *CT Labs: Microsoft Surface Tagged Objects and Tagged Visualizations* [online].
URL:<http://ctlabs.blogspot.com/2009/05/microsoft-surface-tagged-objects-and.html>

[8] *Download details: Byte Tags* [online].
URL:<http://www.microsoft.com/downloads/details.aspx?FamilyID=72BABCDA-465B-4808-86EE-6E23F7967D3E&displaylang=en& displaylang=en >

[9] *XML feed – iShopy.eu* [online].
URL:<http://ishopy.eu/xmlfeed.php>

[10] *Systém.Data.SQLite* [online].
URL:<http://sqlite.phxsoftware.com>

[11] *BBC NEWS | Technology | Second generation Surface coming* [online].
URL:<http://news.bbc.co.uk/2/hi/technology/7945154.stm>

8. Doporučená literatura a odkazy

Při zpracování tohoto tématu bylo použito i literatury, které se problematika Microsoft Surface přímo netýká, byla však užitečná pro doplnění určitých mezer a nejasností. Z těchto zdrojů nebylo citováno, proto se uvádí mimo reference.

- *Začíname s WPF | Vyvojar.cz* [online].
  URL:<http://www.vyvojar.cz/Series/3-zaciname-s-wpf.aspx>

- SELLS CH. – GRIFFITHS I. *Programming WPF : Building Windows UI with Windows Presentation Foundation.* 2nd edition. O'Reilly Media, Inc., 2007. ISBN 0-596-51037-3

- Microsoft Surface
  URL:<http://msdn.microsoft.com/en-us/library/ee804845.aspx>

- *Multi-Touch Systems that I Have Known and Loved* [online].
  URL:<http://www.billbuxton.com/multitouchOverview.html>

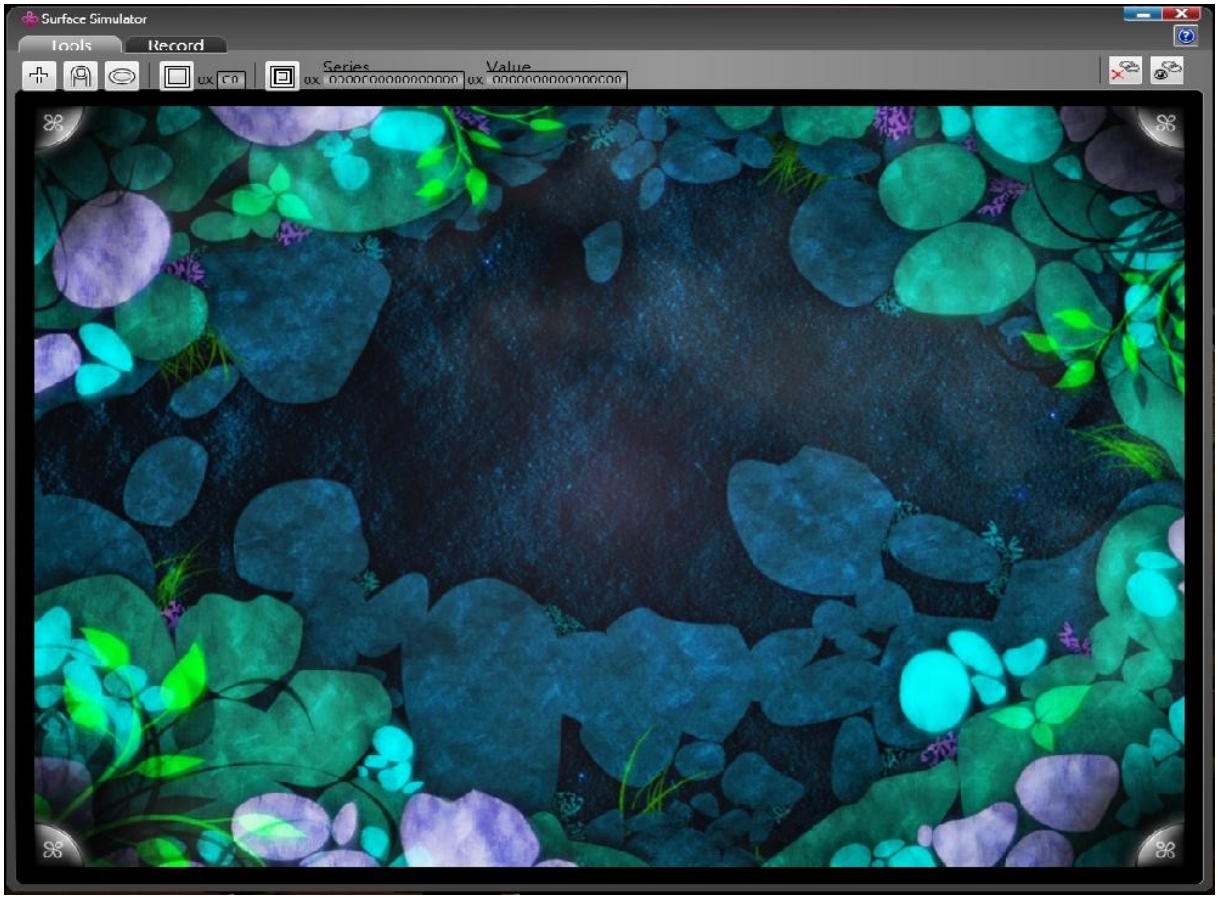
- *Pravidla českého pravopisu | pravidla.cz* [online].
  URL:<http://www.pravidla.cz>

- *Google* [online].
  URL:< http://www.google.cz/>

9. Příloha A - Surface Simulator

Instalováním Microsoft Surface SDK získá uživatel přístup k nástroji, který mu umožňuje simulovat chování aplikace bez nutnosti vlastnit stolek. Tímto nástrojem je Microsoft Surface Simulator. Vývojář, který hodlá s Microsoft Surface pracovat, by měl tento nástroj bezchybně ovládat a vyznat se v jeho funkcích. Vše potřebné najde v této příloze.

Hlavní výhodou tohoto simulátoru spočívá v jeho propojení s Visual C# 2008 (nebo s Visual Studio 2008). Pokud jsou tyto programy spuštěny na stejné úrovni (Administrator nebo User), dojde ke vzniku vazby, která umožňuje spouštět programy v Surface Simulatoru z prostředí Visual Studia. Jakákoliv aplikace, která je ve Visual Studiu spuštěna, je simulátorem prohlédnuta, zda neobsahuje prvky Surface. Pokud je tomu tak, dojde k přesměrování do simulátoru. Výhoda tohoto systému je zřejmá. Surface Simulator lze použít nejen k prohlížení hotového produktu, ale i k odlaďování aplikace.



Obrázek 38: Water attract v Surface Simulator

Hotové aplikace se spouští pomocí přístupových bodů, které se nacházejí v rozích okna. Po kliknutí na libovolný z nich dojde k vypnutí současné aplikace a program simulátoru otevře menu možných aplikací. Pokud došlo k nainstalování vzorových aplikací, objeví se v tomto menu. Bohužel, vyskytují se zde jen aplikace registrované Surface jednotkou a mezi ně nepatří aplikace vytvořené uživateli (důvodem může být skutečnost, že aplikace se nacházejí v jiném adresáři, simulátor je nerozezná a podobně). Pro spuštění vytvořených aplikací lze buď registrovat produkt nebo je spouštět

přes Visual Studio. Je třeba si však uvědomit jednu věc. Simulátor dokáže ovládat jen jednu aplikaci a současné spuštění několika aplikací nebude fungovat.

Během sbírání podkladů pro tuto diplomovou práci bylo zjištěno, že Surface Simulator má vazbu i na jiný produkt téže společnosti: Microsoft Expression Blend. Autor tohoto textu to považuje za přínosné, neboť ruku v ruce s funkčností aplikace jde i vzhled, který se v prostředí Visual Studia vyvíjí dost obtížně. Teď však lze práci na vývoji dělit podle použitého nástroje: Visual Studio používat pro tvorbu funkčního základu aplikace a vzhled dolaďovat v Microsoft Expression Blend.

Samotná aplikace Surface Simulator běží v okně s rozlišením 1024 na 768 pixelů. Při prvním spuštění se objeví aplikace *Water attract*, která má navnadit začínající uživatele. Už na této aplikaci lze vysvětlit pár základních postupů. Jakým způsobem se aplikace ovládá, na to jistě přijde každý uživatel sám. Prostými tahy myši po obrazovce se stisknutým levým tlačítkem lze čeřit pomyslnou hladinu, kterou tato aplikace vytváří. V základním nastavení má ukazatel tvar prstu, jelikož tato aplikace se zaměřuje hlavně na kontakt tohoto typu. Simulator dokonce umožňuje připojit i více myší současně, a tak simulovat pohyb více prstů po displeji.

Pokud není další myš k dispozici, lze použít ještě jeden trik, který simulátor umožňuje. A tím je uzamknutí kontaktu. Provádí se stisknutím levého tlačítka myši a následně pravého tlačítka. Tím dojde k vzniku nového kontaktu s tím, že starý je stále umístěn na povrch stolu. Díky tomu lze i s jednou myší simulovat akce, které vyžadují více kontaktů současně, jako je například zvětšování a zmenšování objektů.

Následující text slouží k tomu, aby si uživatel udělal obrázek o všech jednotlivých funkcích, které tento simulátor nabízí. Pro bližší nahlédnutí doporučuje autor nahlédnout do dokumentace, která je součástí Microsoft Surface SDK.

- Contact selector – slouží k výběru více uzamknutých kontaktů současně. Svým provedením se neliší od běžného selektoru, přístupného v prostředí Windows. Držením levého tlačítka se vytváří obdélníková oblast, do níž se zachytává uživatelem požadovaný výběr. Pomocí selektoru lze upravit pozici a orientaci kontaktu, v případě blobu lze i upravit tvar.

- Finger - simuluje dotek prstem. Čára z něj je široká a oválná, jako otisk prstu. Je to základní kontakt, kterým se ovládá většina aplikací. Kurzor myši má taktéž symbol prstu a používání levého tlačítka určuje kontakt. Pravé tlačítko myši slouží k uzamknutí kontaktu na plochu. Ukazateli lze pomocí středového kolečka myši určovat orientaci, což ještě víc podporuje simulování přirozeného používání aplikace.

- Blob – simuluje libovolný objekt elipsového tvaru. Tvar blobu lze upravovat. Blob o malém poloměru může například simulovat pero či jiný psací nástroj. Platí pro něj stejné možnosti jako pro Finger.

- Byte tag - velmi důležitá funkce. Protože simulátor postrádá prostředky pro rozpoznávání objektů, je tato funkce nahrazena tímto prvkem. Při kontaktu typu tag je aplikaci předána událost, že byl rozpoznán Byte tag s příslušnou hexadecimální hodnotou. Jedná se o jeden ze dvou typů, které Microsoft Surface ovládá.

- Identity tag – druhý typ tagů v prostředí Microsoft Surface. Na rozdíl, od jeho předchůdce, tento tag je identifikován nejen hexadecimálním číslem hodnoty, ale i číslem série, což ho v prostředí Surface dělá jedinečným.

- Clear all contacts – funkce, jež vymaže všechny uzamknuté kontakty na ploše.

- Hide contacts – skryje všechny kontakty. Ty lze stále přidávat i odebírat, nejsou však v prostředí simulátoru nijak zviditelněny. Není vidět ani kurzor myši. Tato funkce má jenom jediný účel zvýšit v uživateli dojem, že pracuje se samotným stolkem Microsoft Surface a ne s pouhým simulátorem.

- Acess point – tato funkce umožňuje přístup do menu, v kterém lze vybírat jiné aplikace. Také slouží k změně orientace aplikace. Proto je dobré si v simulátoru dávat pozor na to, který z přístupových bodů se používá. V případě, že je použit jeden ze dvou na obrazovce nahoře umístěných přístupových bodů, dojde k obrácení orientace aplikace o 180 stupňů. Děje se tak na základě věrné simulace stolku, kde se vlivem větších rozměrů předpokládá, že uživatel, jež tlačítko stisknul, stojí na opačné straně.

- Recording – pouze pro účely simulátoru, v prostředí stolku se tato funkce vůbec nevyskytuje. Po spuštění dojde k nahrávání všech kontaktů, které se na ploše simulátoru odehrály. Nejedná se však o video, ale o aktivní nahrání všech pohybů, které uživatel učinil. Při přehrávání dojde k provedení všech pohybů a kontaktů, jak byly nahrány. Ty však nejsou vázány na aplikaci, takže lze vyměnit aplikaci za některou jinou a přehrát si kontakty na ní.

Tímto byl nástroj Microsoft Surface Simulator do základu popsán. Poskytuje neocenitelné služby všem vývojářem aplikací Microsoft Surface a jeho použití je dostatečně jednoduché, aby ho pochopil i začátečník.

10. Příloha B – Obsah CD

- Ukázkové aplikace
    - Aplikace Chaos
        - Zdrojové kódy
        - Spustitelné kódy
    - Aplikace Tags
        - Zdrojové kódy
        - Spustitelné kódy
    - Aplikace SurfaceShop
        - Zdrojové kódy
        - Spustitelné kódy
- Tento dokument (DOCX a PDF)
- Obrázky použité v tomto dokumentu
- Odkazy na některé zajímavé prezentace Microsoft Surface (DOCX a PDF)